BRNO UNIVERSITY OF TECHNOLOGY

Faculty of Electrical Engineering
and Communication

MASTER'S THESIS

Brno, 2021 Bc. Veronika Poláková

BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF BIOMEDICAL ENGINEERING
ÚSTAV BIOMEDICÍNSKÉHO INŽENÝRSTVÍ

UTILIZATION OF CONVOLUTIONAL NEURAL NETWORKS FOR SEGMENTATION OF MOUSE EMBRYOS CARTILAGINOUS TISSUE IN MICRO-CT DATA

VYUŽITÍ KONVOLUČNÍCH NEURONOVÝCH SÍTÍ PRO SEGMENTACI CHRUPAVČITÝCH TKÁNÍ MYŠÍCH EMBRYÍ V MIKRO-CT DATECH

MASTER'S THESIS
DIPLOMOVÁ PRÁCE

AUTHOR — **Bc. Veronika Poláková**
AUTOR PRÁCE

SUPERVISOR — **Ing. Jiří Chmelík, Ph.D.**
VEDOUCÍ PRÁCE

BRNO 2021

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

Master's Thesis

Master's study program **Biomedical Engineering and Bioinformatics**

Department of Biomedical Engineering

Student: Bc. Veronika Poláková ***ID:*** 191656

Year of study: 2 ***Academic year:*** 2020/21

TITLE OF THESIS:

Utilization of convolutional neural networks for segmentation of mouse embryos cartilaginous tissue in micro-CT data

INSTRUCTION:

1) Get acquainted with a principle of X-ray micro-computed tomography (CT) and with its usability for investigation of biological tissue samples. 2) Carry out a search of literature dealing with convolutional neural networks (CNN) and their application for 3D image data segmentation. 3) Based on the previous search, propose a concept of computer program dealing with the segmentation of craniofacial cartilaginous tissue of mouse embryos acquired by the micro-CT system. 4) In the appropriately chosen programming framework, implement the proposed CNN together with a suitable data pre-processing. Train the implemented CNN on the available image dataset. 5) Optimize the proposed approach in terms of the maximisation of its segmentation quality. 6) Statistically evaluate the functionality of the proposed method and appropriately discuss the achieved results.

RECOMMENDED LITERATURE:

[1] ASGARI TAGHANAKI, Saeid, Kumar ABHISHEK, Joseph Paul COHEN, Julien COHEN-ADAD a Ghassan HAMARNEH. Deep semantic segmentation of natural and medical images: a review. Artificial Intelligence Review. DOI: 10.1007/s10462-020-09854-1. ISSN 0269-2821.

[2] TESAŘOVÁ, M., T. ZIKMUND, M. KAUCKÁ, I. ADAMEYKO, J. JAROŠ, D. PALOUŠEK, D. ŠKAROUPKA a J. KAISER. Use of micro computed-tomography and 3D printing for reverse engineering of mouse embryo nasal capsule. Journal of Instrumentation. 2016, 11(03), C03006-C03006. ISSN 1748-0221.

Date of project specification: 8.2.2021 ***Deadline for submission:*** 21.5.2021

Supervisor: Ing. Jiří Chmelík, Ph.D.
Consultant: Ing. Jan Matula

prof. Ing. Ivo Provazník, Ph.D.
Chair of study program board

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní program **Biomedicínské inženýrství a bioinformatika**

Ústav biomedicínského inženýrství

Studentka: Bc. Veronika Poláková ***ID:*** 191656
Ročník: 2 ***Akademický rok:*** 2020/21

NÁZEV TÉMATU:

Využití konvolučních neuronových sítí pro segmentaci chrupavčitých tkání myších embryí v mikro-CT datech

POKYNY PRO VYPRACOVÁNÍ:

1) Seznamte se s principem rentgenové počítačové mikrotomografie (CT) a jejím využitím pro studium biologických tkání vzorků. 2) Vypracujte literární rešerši zabývající se konvolučními neuronovými sítěmi (CNN) a jejich aplikací pro segmentaci 3D obrazových dat. 3) Na základě provedené rešerše navrhněte koncepci programu pro segmentaci chrupavčitých tkání obličejové části myších embryí v obrazech pořízených s využitím mikro-CT. 4) Ve vhodně zvoleném programovacím jazyce implementujte vybranou CNN společně s implementací vhodně zvoleného předzpracování dat. Implementovanou síť trénujte na dostupné databázi obrazových dat. 5) Navržený postup optimalizujte z hlediska maximalizace úspěšnosti segmentace. 6) Statisticky zhodnoťte funkčnost navržené metody a dosažené výsledky vhodně diskutujte.

DOPORUČENÁ LITERATURA:

[1] ASGARI TAGHANAKI, Saeid, Kumar ABHISHEK, Joseph Paul COHEN, Julien COHEN-ADAD a Ghassan HAMARNEH. Deep semantic segmentation of natural and medical images: a review. Artificial Intelligence Review. DOI: 10.1007/s10462-020-09854-1. ISSN 0269-2821.

[2] TESAŘOVÁ, M., T. ZIKMUND, M. KAUCKÁ, I. ADAMEYKO, J. JAROŠ, D. PALOUŠEK, D. ŠKAROUPKA a J. KAISER. Use of micro computed-tomography and 3D printing for reverse engineering of mouse embryo nasal capsule. Journal of Instrumentation. 2016, 11(03), C03006-C03006. ISSN 1748-0221.

Termín zadání: 8.2.2021 ***Termín odevzdání:*** 21.5.2021

Vedoucí práce: Ing. Jiří Chmelík, Ph.D.
Konzultant: Ing. Jan Matula

prof. Ing. Ivo Provazník, Ph.D.
předseda rady studijního programu

ABSTRACT

Automatic segmentation of the biological structures in micro-CT data is still a challenge since the object of interest (craniofacial cartilage in our case) is commonly not characterized by unique voxel intensity or sharp borders. In recent years, convolutional neural networks (CNNs) have become exceedingly popular in many areas of computer vision. Specifically, for biomedical image segmentation problems, U-Net architecture is widely used. However, in the case of micro-CT data, there is a question whether 3D CNN would not be more beneficial. The master thesis introduced CNN architecture based on V-Net as well as the methodology for data preprocessing and postprocessing. The baseline architecture was further optimized using advanced architectural modifications such as Atrous Spatial Pyramid Pooling (ASPP) module, Scaled Exponential Linear Unit (SELU) activation function, multi-output supervision and Dense blocks. For network learning, modern approaches were used including learning rate warmup or AdamW optimizer. Even though the 3D CNN do not outperform U-Net regarding the craniofacial cartilage segmentation, the optimization raises the median of Dice coefficient from 69.74 % to 80.01 %. Therefore, utilizing these advanced architectural modifications is highly encouraged as they can be easily added to any U-Net-like architecture and may remarkably improve the results.

KEYWORDS

convolutional neural networks, X-ray micro computed tomography, image segmentation, cartilaginous tissue, V-Net

ABSTRAKT

Automatická segmentace biologických struktur v mikro-CT datech je stále výzvou, protože často objekt zájmu (v našem případě obličejová chrupavka) není charakterizovaný unikátním jasem či ostrými hranicemi. V posledních letech se konvoluční neuronové sítě (CNNs) staly mimořádně populárními v mnoha oblastech počítačového vidění. Konkrétně pro segmentaci biomedicínských obrazů je široce používaná architektura U-Net. Nicméně v případě mikro-CT dat vyvstává otázka, zda by nebylo výhodnější použít 3D CNN. Diplomová práce navrhla CNN architekturu založenou na síti V-Net včetně metodologie pro předzpracování a postprocessing dat. Základní architektura byla dále optimalizována pomocí pokročilých architektonických modifikací jako jsou pyramidální modul dilatovaných konvolucí (ASPP modul), škálovatelná exponenciálně-lineární jednotka (SELU aktivační funkce), víceúrovňová kontrola učení (multi-output supervision) či bloky s hustými propojeními (Dense blocks). Pro učení sítě byly použity moderní přístupy jako zahřívání kroku učení (learning rate warmup) či AdamW optimalizátor. I přes to, že 3D CNN v úloze segmentace obličejové chrupavky nepřekonala U-Net, optimalizace zvýšila medián Dice koeficientu z 69,74 % na 80,01 %. Používání těchto pokročilých architektonických modifikací v dalším výzkumu je proto vřele doporučováno, jelikož můžou být přidány do libovolné architektury typu U-Net a zároveň výrazně zlepšit výsledky.

KLÍČOVÁ SLOVA

konvoluční neuronové sítě, rentgenová výpočetní mikrotomografie, segmentace obrazu, chrupavčitá tkáň, V-Net

POLÁKOVÁ, Veronika. *Utilization of convolutional neural networks for segmentation of mouse embryos cartilaginous tissue in micro-CT data*. Brno, 2021, 73 p. Master's Thesis. Brno University of Technology, Faculty of Electrical Engineering and Communication, Department of Biomedical Engineering. Supervised by Ing. Jiří Chmelík, Ph.D.

Typeset by the `thesis` package, version 4.00; `http://latex.feec.vutbr.cz`

ROZŠÍŘENÝ ABSTRAKT

Úvod

Konvoluční neuronové sítě (z angl. *convolutional neural networks*, CNNs) jsou v posledních letech široce používané při řešení různých problémů počítačového vidění včetně segmentace obrazu. Jejich hlavní výhodou je schopnost naučit se extrahovat relevantní příznaky pro daný problém, a proto jsou vysoce vhodné pro komplexní problémy, kde je manuální extrakce příznaků velmi náročná. Takovým problémem je i segmentace obličejové chrupavky myších embryí v mikro-CT datech.

Rentgenová výpočetní mikrotomografie (nebo zkráceně mikro-CT) je zobrazovací modalita umožňující studium vnitřní struktury vzorku (v našem případě embrya) v řádech mikrometrů. S využitím mikro-CT dat lze vytvořit 3D modely obličejové chrupavky embryí, které jsou dále používány biology pro výzkum vývoje obličeje savců. Například srovnání těchto modelů mezi kontrolními embryi a mutanty může pomoci objasnit vliv jednotlivých genů na vývoj obličeje. Pro vytvoření modelů je nicméně zapotřebí nejdříve obličejovou chrupavku z mikro-CT dat vysegmentovat. Manuální segmentace je však velmi časově náročná a rovněž závislá na operátorovi, a proto existuje poptávka po automatickém přístupu.

V předešlém výzkumu bylo na daný segmentační problém vyzkoušeno několik metod klasického zpracování obrazu i strojového učení, nicméně tyto metody byly zcela překonány přístupem hlubokého učení (U-Net) [42]. Tyto výsledky byly motivací pro větší výzkum v oblasti konvolučních neuronových sítí a pro vznik této práce. Konkrétněji se diplomová práce zabývá otázkou využitelnosti 3D CNNs na daný segmentační problém, přičemž navrhuje architekturu založenou na síti V-Net včetně metodologie pro preprocessing a postprocessing dat. Velký důraz je rovněž kladen na optimalizaci sítě se zaměřením na využití pokročilých architektonických modifikací převzatých ze soudobé literatury.

Popis řešení

Teoretická část práce začíná bližším představením mikro-CT, tedy modality použité pro sběr dostupných dat. Rovněž jsou diskutovány problémy spojené se zobrazováním *biologických* vzorků, které umožní čtenáři lépe pochopit problematičnost automatické segmentace v mikro-CT datech. Rozsáhlá kapitola je věnována CNNs a procesu jejich učení. Podrobně jsou rozebrány problémy, které můžou při učení i mimo něj nastat, a jsou představena řešení těchto problémů dostupná v literatuře. Na základě provedené literární rešerše jsou nakonec popsány i možné přístupy segmentace objemových dat pomocí CNNs a dále milníkové architektury pro sémantickou segmentaci.

V praktické části práce jsou pak představena dostupná data a je dále diskutováno začlenění 3D informace do segmentačního procesu. Konkrétněji se přistupuje k 3D segmentaci po částech, jelikož obličejová chrupavka je podlouhlá ve všech třech osách a nelze ji tedy ve 3D segmentovat najednou kvůli paměťovým limitacím dostupného hardwaru. Proces 3D segmentace po částech zahrnuje vhodné předzpracování i postprocessing dat. Zásadní částí práce je pak navržení 3D CNN architektury a její optimalizace.

Navržená architektura byla inspirována sítí V-Net, která byla dále modifikována pro daný segmentační problém. Hyperparametry učení byly nastaveny na základě moderních poznatků v současné literatuře. Důraz byl kladen na následnou optimalizaci navržené architektury a to konkrétně pomocí inkorporace pokročilých architektonických modifikací vycházejících z literární rešerše.

Proces optimalizace lze popsat následovně:

1) Do poslední úrovně CNN architektury byl vložen pyramidální modul dilatovaných konvolucí (ASPP modul [12]), který umožnil síti extrahovat více kontextuální informace.
2) Pro potlačení problému vnitřního posunu rozložení příznaků během učení (angl. *internal covariate shift problem* [25]) byla použita škálovatelná exponenciálně-lineární jednotka (z angl. *Scaled Exponential Linear Unit*, SELU [32]). Tato aktivační funkce má samonormalizující vlastnost, díky které je možné držet rozložení příznaků v určitém rozmezí a vyhnout se tak zmíněnému problému, který znesnadňuje učení. Výhodou SELU oproti jiným normalizačním technikách je absence nastavitelných hyperparametrů a nezávislost na velikosti mini-batche.
3) Pro zajištění více stabilního učení a vysoké kvality příznaků v celé hloubce sítě bylo učení závislé na chybové funkci ve všech úrovních CNN architektury. Aby byla kompenzována různá velikost segmentačních predikcí v každé úrovni, celková chyba učení byla spočítána jako *váhovaný* průměr jednotlivých chyb.
4) Reziduální bloky sítě byly nahrazeny hustě propojenými bloky (z angl. *Dense blocks* [26]), aby byly již extrahované příznaky co nejvíce využity.

Výsledky a závěr

Pro vyhodnocení úspěšnosti segmentace byla použita křížová validace. Medián Dice koeficientu základního modelu činil 69,74 %, přičemž tato hodnota byla optimalizací krok po kroku zlepšována až na 80,01 %. Optimalizací se podařilo nejen zvýšit medián Dice koeficientu, ale rovněž snížit mezikvartilové rozpětí výsledků a zpřesnit tak odhad výkonnosti modelu.

Dále bylo provedeno srovnání navrženého modelu pro 3D segmentaci po částech s U-Netem z předchozího výzkumu [42]. U-Net dosahoval na stejném datasetu

mediánového Dice koeficientu 85,09 %, a proto lze konstatovat, že pro segmentaci obličejové chrupavky se zdá být důležitější celková anatomická informace v řezu než přidání okolních řezů.

Závěrem lze tedy říci, že byl navržen postup 3D segmentace po částech zahrnující vhodné předzpracování dat, zpracování 3D CNN a fázi postprocessingu. Na základě literární rešerše byla vytvořena 3D CNN architektura typu V-Net spolu s popisem procesu učení. Důraz byl kladen na optimalizaci navržené sítě skrze novodobé architektonické modifikace, které zásadním způsobem zlepšily její výkonnost (69,74 % $\rightarrow$ 80,01 %). Rovněž byl učiněn závěr, že na segmentaci podlouhlých struktur (jako je obličejová chrupavka) je výhodnější použít segmentaci řez po řezu než 3D segmentaci po částech. Navržený optimalizační proces může být jednoduše implementovaný do U-Netu z předchozího výzkumu, což by zřejmě vedlo ke zlepšení zatím nejlepších dosažených výsledků pro daný segmentační problém.

DECLARATION

I declare that I have written the Master's Thesis titled "Utilization of convolutional neural networks for segmentation of mouse embryos cartilaginous tissue in micro-CT data" independently, under the guidance of the advisor and using exclusively the technical references and other sources of information cited in the thesis and listed in the comprehensive bibliography at the end of the thesis.

As the author I furthermore declare that, with respect to the creation of this Master's Thesis, I have not infringed any copyright or violated anyone's personal and/or ownership rights. In this context, I am fully aware of the consequences of breaking Regulation § 11 of the Copyright Act No. 121/2000 Coll. of the Czech Republic, as amended, and of any breach of rights related to intellectual property or introduced within amendments to relevant Acts such as the Intellectual Property Act or the Criminal Code, Act No. 40/2009 Coll., Section 2, Head VI, Part 4.

Brno ....

author's signature

ACKNOWLEDGEMENT

I would like to thank my supervisor Ing. Jiří Chmelík, Ph.D. for his kind guidance during the elaboration of the thesis. Also, I would like to thank my consultant Ing. Jan Matula for introducing me into the problem and being there for me whenever needed. Next, I would like to thank the research group leader prof. Ing. Jozef Kaiser, Ph.D. and the laboratory leader Ing. Tomáš Zikmund, Ph.D. for accepting me in the team of undergraduate students in CTlab CEITEC BUT and also for enabling me to work on the laboratory computer equipped with professional hardware that was crucial for the thesis emergence. V neposlední řadě bych ráda poděkovala své rodině za podporu po celou dobu studia.

Contents

List of Figures

List of Tables

Introduction

In recent years, convolutional neural networks (CNNs) have become exceedingly popular in computer vision tasks such as image classification, object detection, or segmentation. Since the network itself is able to learn relevant features for the given task, CNN is well-suited for complex problems where the manual feature extraction would be too demanding. An example of such a complicated structure is a mouse embryo's craniofacial cartilage which is the object of interest in the master thesis.

A mouse is a typical animal model for studying the biological processes of mammals, including humans. Developmental biologists use mouse embryos to investigate the complex process of morphogenesis but also an abnormal growth (dysmorphogenesis) of the face [9, 28, 29]. Various visualization techniques are used for this task, including X-ray micro computed tomography (micro-CT) [28, 29, 58]. Using this imaging technique, 3D image data of a mouse embryo are obtained and used to visualize the craniofacial cartilage either in different planes or as a 3D model. This visualization enables the developmental biologists get a better intuition about the overall shape of the structure and compare the craniofacial cartilage in the various developmental stages of the mouse more easily. Moreover, the craniofacial cartilage can also be compared between a wild type and a mutant mouse which can reveal the effect of the particular genes on the face development [9, 28, 29].

In order to create the 3D model, craniofacial cartilage has to be segmented in the 3D image data. Manual segmentation is a very time-consuming and operator dependent process, therefore, there is a demand for an automatic approach. Several image processing (region growing with adaptive threshold, morphological geodesic active contours) and machine learning approaches (manual feature extraction with random forest classifier) were already examined for solving this issue. Nevertheless, they were entirely surpassed by a deep learning approach (U-Net) [42], encouraging further research in the area of convolutional neural networks (CNNs).

A question that arises considering volumetric data is whether using a 3D approach could improve the segmentation results. The master thesis focuses on this issue and proposes a 3D solution suitable for the given segmentation problem. Specifically, in the theoretical part, the reader is acquainted with imaging biological samples using micro-CT and especially with CNNs and their utilization in 3D image data segmentation. In the practical part, according to the available dataset, the solution proposal is presented, including data preprocessing, processing by CNN and postprocessing. The subsequent CNN optimization process is also thoroughly described with an emphasis on the utilization of advanced architectural modifications presented in the literature in recent years. Finally, the results are evaluated and discussed.

1 Imaging biological samples using X-ray micro computed tomography

X-ray micro computed tomography (X-ray microtomography, micro-CT) is an imaging technique that allows us to study a sample in a micrometer range. As we can also see the sample's inner structure without destructing it, micro-CT has become popular in many areas including biology [16, 46, 58].

In micro-CT measurement, the sample is situated between the two main components of the imaging system – X-ray tube and detector. The closer the sample is placed to the X-ray tube, the greater the magnification is, but also the smaller field of view is. Since the micro-CT detector contains many (ordinarily units of megapixels) small detection elements [34], the greater magnification results in a better spatial resolution of the sample (the best micro-CT systems are able to achieve a voxel size up to 1 µm) [55]. The first step of the tomography algorithm is a decomposition of the 3D scene into a set of 2D projections obtained from many different angles. In order to accomplish this, micro-CT typically rotates the sample (in contrast to medical CT devices, where the X-ray tube and detector rotate) [34]. More specifically, the sample is placed on a holder made of low-density material (typically plastic or glass) which is mounted on a rotating stage [16].

After the sample is properly positioned, the measurement itself can start. Firstly, according to the current and voltage settings, the X-ray tube generates given amount of the X-ray photons in a specific energy range. Microfocus X-ray tube is used in micro-CT to avoid decreasing a spatial resolution because of penumbra [22]. Next, a filter made from a particular material (usually copper, tin or aluminium [16]) is commonly placed just after the X-ray tube so that the low-energy photons, which the sample would completely absorb, are filtered [22]. Then, in some geometries, where the source itself does not produce the desired shape of the X-ray beam, a collimator is added [55]. After that, X-rays finally reach the sample, where each photon is attenuated according to Beer-Lambert law [22].

According to the law, there is a dependency between attenuation and the material (tissue in the biological case) characteristics which is the cornerstone of X-ray imaging. Specifically, mass attenuation coefficient (given by the elemental composition of the material and X-ray beam energy), density, and thickness of the material are important quantities of the sample [22]. The effect of material thickness is suppressed by a tomography algorithm, which is a big advantage of tomography systems over the planar ones.

After passing through the sample, the X-ray beam is detected by the detector performing a conversion of light into electricity. In some geometries, the col-

limator is also placed between the sample and the detector to shield the detector from scattered photons [55]. Finally, a projection image is computed from the incident and transmitted X-ray beam intensity. In the following step, the sample is slightly rotated, and the next projection image is measured.

The projection images are used as the input for a tomographic reconstruction algorithm usually based on a filtered backprojection or an iterative approach [22]. Finally, the reconstruction algorithm yields the 3D image data, where the value of each voxel represents the X-ray attenuation in a given region of the sample.

Imaging of biological samples using X-ray microtomography may be challenging for two main reasons. Firstly, we are often interested in structures which attenuate X-rays similarly [46]. Secondly, even *ex vivo* biological samples tend to move (because of shrinkage due to dehydratation) which, especially in longer scan times[1], results in blurred 3D image data [16]. Both problems are addressed with correct prescanning sample preparation.

Regarding *in vivo* scanning, micro-CT with a rotating gantry (where X-ray tube and detector are placed) is used instead of systems with a rotating sample. To reduce movement artifacts the subject is under anesthesia, and scanning is gated according to a cardiac or a respiratory signal [3]. Since contrasting soft tissues is really challenging in a living organism, typical applications of *in vivo* scanning focus on structures with good natural contrast (bones, lungs, specific tumors [3]). Additionally, a contrast agent (based on iodine) can be infused into the circulatory system, which enables examining vessels and heart [3].

As far as the *ex vivo* biological samples are concerned, a staining procedure is commonly applied to them. Since the various tissues absorb the staining solution differently, the contrast between these tissues is enhanced. The contrast agent must have suitable attenuation properties so that the contrast in a scene is also observable in an X-ray image. Typically, a staining solution contains an element with a high atomic number, e.g. iodine, osmium, gold, silver, tungsten, etc. [46]. Besides the contrast agent, other chemicals (depending on the protocol) are added to the staining solution to enhance a diffusion of the contrast agent into the sample. Finally, the sample is placed either into a thin holder with liquid media [43] or into an embedding (e.g. resin [43, 46] or agarose gel [58]) to avoid any movement during the scanning.

[1]Since getting a better signal-to-noise ratio (SNR) is more advantageous than a small dose in measuring *ex vivo* samples, the measurement itself may take dozens of minutes.

2 Application of convolutional neural networks in 3D image data segmentation

Segmentation is a process of dividing the image data into segments. In other words, the aim of the segmentation is to classify each voxel to the right class (craniofacial cartilage or everything else (background) in our case). Two subcategories of segmentation exist in a computer vision – semantic segmentation and instance segmentation [36]. Since the whole craniofacial cartilage is regarded as a single instance, semantic segmentation is used in the master thesis.

Convolutional neural networks (CNNs) are considered as the neural networks well-suited for grid-like data processing [19]. Since the 3D image data can be viewed either as a stack of pixel grids (2D images) or directly as a voxel grid, CNNs are commonly used for processing them. Specifically, CNNs use a set of kernels extracting relevant features for a given object of interest. In contrast to the artificial neural networks (ANNs), CNNs respect that voxels close together are more related than the distant ones which brings important advantages. Firstly, convolving the image data with a given kernel reveals the feature in any coordinates of the image data meaning that various translations of the object of interest do not affect the performance of the CNN [19]. Secondly, we can computationally profit from using local operators (kernels) since a weight matrix of each layer is sparse (the distant voxels have zero weights), and the number of learnable parameters is reduced due to parameter sharing (the same kernel is used in various coordinates) [19].

2.1 Building blocks of CNNs

CNN architecture is defined by its building blocks, which may be placed arbitrarily sequentially or parallel to each other. The only condition is that the output of a given block must have the same dimension as the input of the following one. To avoid changes in dimensions due to convolution with a given kernel, the input may be padded by zeros. The output size of the block can also be affected by the stride argument that defines sliding of the kernel over the input. Although various architectures may include their own specialized blocks, the following blocks are the basis of *segmentation* CNNs.

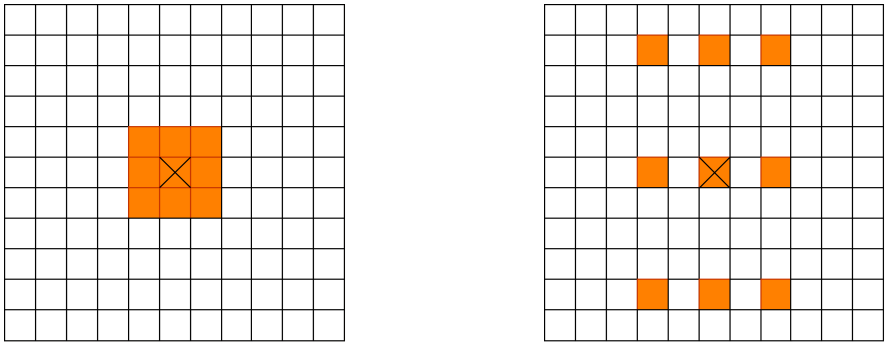
Convolutional layer. The layer consists of a defined number of kernels extracting the features locally in space when using all channels. This implies that kernels in 3D CNNs are actually 4D (height, width, depth, channels), whereas 3D kernels (height, width, channels) can be found in 2D CNNs. The kernel size is also user-

defined, however, common sizes are 3 or 5 in each spatial dimension as the bigger sizes are computationally demanding.

In the case of a bigger receptive field requirement, multiple attitudes can be seen in the literature. Firstly, the layer with a large-sized kernel may be expressed as a stack of layers with small-sized kernels resulting in the same receptive field, but the number of learnable parameters is reduced, implying quicker training and a network less prone to overfitting [54]. However, the depth of the network is enlarged meaning that more activation maps have to be stored in the memory which may be seen as a disadvantage of this attitude. Secondly, the dilated convolution approach can be applied using a small-sized kernel on the input with gaps defined by a dilation factor [62]. Since some input elements are skipped (Figure 2.1), the receptive field is enlarged, whereas the number of learnable parameters persists small and also the depth of the network remains unchanged using the dilated approach. Lastly, the input can be downsampled before applying the kernel, however, this attitude cannot be used very frequently because of losing a part of the information by each downsampling.

Kernels of size 1 in each spatial dimension are commonly utilized for changing the number of channels in the feature space. Typically, a lot of features are linearly combined into a small number of features in this manner so that subsequent calculations are not so computationally demanding.

Because of practical reasons, the convolutional layer is implemented as a set of cross-correlations instead of convolutions [19]. In other words, we do not bother with the kernel flipping and use directly an element-wise multiplication of the input and the kernel followed by summation. In practice, this is implemented as a matrix multiplication of the vectorized kernel and the vectorized input.



(a) dilation rate = (1, 1)

(b) dilation rate = (4, 2)

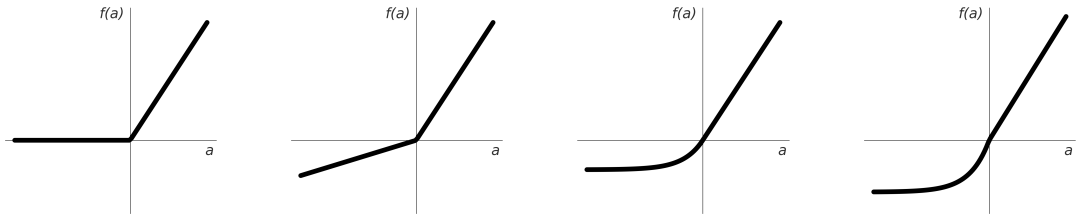
Figure 2.1: Comparison of convolutions with various dilation rates. The result of convolution in a given position (cross) is obtained using specific pixels of the input image (orange). If the dilation rate is equal to one in all dimensions, the dilated convolution can be considered as the "classic" convolution.

Activation layer. Deep linear models do not bring additional information into the problem compared to a single linear unit since a stack of linear combinations can be expressed as a single linear combination. This is the reason why it is crucial to incorporate nonlinearity into the model using the activation layer. Commonly, activation functions contain nonlinearity for the input values close to zero. Therefore, data standardization is a necessary step before the network's learning and processing. Furthermore, the derivative of the activation function must be defined for its whole domain so that the parameters can be updated during learning properly.

Sigmoidal or softmax activation functions are commonly used in the final layer of CNNs to determine the probability of each spatial element belonging to a given class. In the past, the sigmoidal or tanh activation functions were commonly used throughout the network, however, nowadays their usage is deprecated because of a significant vanishing gradient problem [19].

Today, the state of the art activation function is rectified linear unit (ReLU) due to its simplicity and good performance in deep networks. Nevertheless, many extensions of ReLU (Figure 2.2) exist in the literature addressing its drawbacks such as nonzero mean or so-called dying ReLU problem (characterized by inactive neurons unable to learn and outputting zero for any input) [7]. For instance, Leaky ReLU [41] mitigates the dying ReLU problem due to the nonzero slope in the negative domain of the function. In other words, neurons outputting negative activations can be still incorporated into the learning process and become important feature extractors at the end of the learning. Parametric ReLU (PReLU) [21] goes even further and defines the already mentioned slope as a learnable parameter. Exponential linear unit (ELU) [15] returns the saturation region in the design of the activation function, however, in comparison with ReLU, the function saturates for more negative inputs and also the level of saturation is negative pushing the output's mean closer to zero. The authors claim that using the saturation region in the activation function is advantageous since the CNN performance is less dependent on the specific content of the *object-free* inputs. ELU afterwards inspired scaled exponential linear unit (SELU) [32] designed to have a so-called self-normalizing property mitigating the internal covariate shift problem as will be described in section 2.2.

Pooling layer. The aim of the layer is an information summarization in each channel separately using a kernel with a predefined behaviour. Usually, the kernel searches for a maximum or an average from local non-overlapping neighborhoods. The pooling operation leads to reduction in spatial dimensions which results in lower computational demands in the following layers. Also, the information summarization may be considered as a form of regularization preventing overfitting.



(a) ReLU (b) PReLU (c) ELU (d) SELU

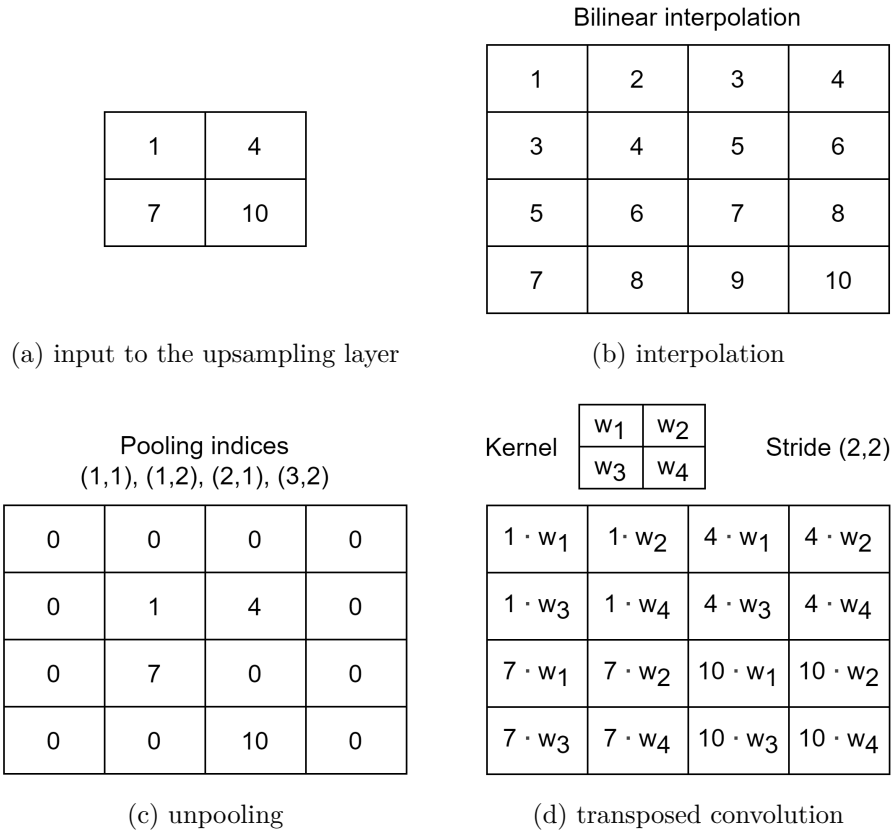
Figure 2.2: ReLU and some of its extensions.

Upsampling layers. There are several upsampling strategies including interpolation, unpooling, or transposed convolution. These layers are necessary to obtain a segmentation image with the same resolution as the input image has. There are no general rules for choosing the right upsampling layer since all of them have advantages as well as disadvantages (Table 2.1). The principle of these layers is depicted in Figure 2.3.

The CNN architecture is typically built by stacking several convolutional and activation layers followed by the layers changing a spatial resolution. Regarding the segmentation CNNs, there are commonly two parts – a contracting part (encoder) and an expanding part (decoder). In the encoder, the input is commonly reduced in spatial dimensions, but enlarged in channel (feature) dimensions. Conversely, the decoder part aims to transform the feature-rich information from reduced spatial resolution to the original one so that the segmentation prediction for the input image can be obtained.

Table 2.1: Advantages and disadvantages of upsampling layers.

Interpolation	+ no additional parameters to learn − the most basic form of upsampling
Unpooling	+ reverse operation to pooling + no additional parameters to learn − pooling indices must be stored in a memory
Transposed convolution	+ learnable upsampling − additional parameters to learn



(a) input to the upsampling layer

(b) interpolation

(c) unpooling

(d) transposed convolution

Figure 2.3: Upsampling layers. Several approaches (b, c, d) exist to upsample a downsampled feature map (a). Note: the hyperparameters (e.g. order of interpolation, kernel size and its stride) are tunable.

2.2 Learning of CNNs

CNN model is defined by its learnable parameters and user-defined hyperparameters. The architecture itself can be considered as a hyperparameter. The way of learning is also user-defined (e.g. optimization algorithm, initial learning rate, scheduling learning rate, etc.). The parameters, e.g. weights and biases of neurons, are firstly initialized randomly and then they are optimized so that the output of CNN matches the ground truth as much as possible. This process is known as learning and comprises of four steps that are repeated over and over during the training phase:

1) *Forward propagation.* The output of the CNN model is computed from the input data.
2) *Loss computation.* The output of the CNN model is compared with the ground truth according to the user-defined loss function. The bigger is difference between the output and the ground truth, the bigger is loss.

3) *Backpropagation.* For each learnable parameter, the gradient of the loss function according to the parameter is computed. In other words, information about how much each parameter affects the loss is obtained.
4) *Update of parameters.* The value of a given learnable parameter is updated according to the gradients to get closer to the global minimum of the loss function.

The speed or even the ability of CNN to learn properly may be affected by several problems. For instance, if the network is very deep, the backpropagated gradients may vanish or explode [19]. Vanishing leads to a situation when the backpropagated gradients are getting smaller and smaller (due to chain rule calculation) meaning that parameters at the beginning of the network do not learn properly. Exploding may be considered as the opposite of vanishing, in other words, the backpropagated gradients are getting bigger and bigger which may lead to unstable learning since the subsequent movement in an optimization landscape is too big. The vanishing or exploding problem can also be seen during the forward propagation in the case of deep networks. For example, if the weights are initialized as too small, the output of particular layers becomes smaller and smaller until it finally vanishes. The speed of learning can also be greatly affected by a so-called internal covariate shift [25]. This phenomenon means that the input data distribution to a given layer varies greatly during the training (because of parameters changes in the previous layers) resulting in a slower optimization of the layer's parameters. Another example of problematic learning is a situation when the parameters have such values that the training gets trapped in local minima or in small-gradient areas far from the global optimum. This implies poor performance or extremely slow speed of learning.

To address these problems, several techniques were developed. For instance, to avoid vanishing or exploding of the information during the first forward propagation as well as during the first backpropagation, several smart random initializations were proposed to preserve data distribution of the input and output throughout the network. The most popular initialization schemes are Glorot (Xavier) initialization [18], frequently used in networks with tanh or sigmoidal activations, and He initialization [21], recommended for networks with ReLU activations. To deal with the vanishing or exploding problem also in the next training iterations, the norm of gradients may be heuristically scaled or clipped [7]. This approach is based on the idea that the main role of the gradient is not defining the step size but the direction of movement which is also preserved by norming [19].

A great impact on improving learning have so-called residual blocks presented in [20]. In addition to a branch with several layers, the residual blocks also include an identity branch that preserves the flowing of the gradient meaning that also

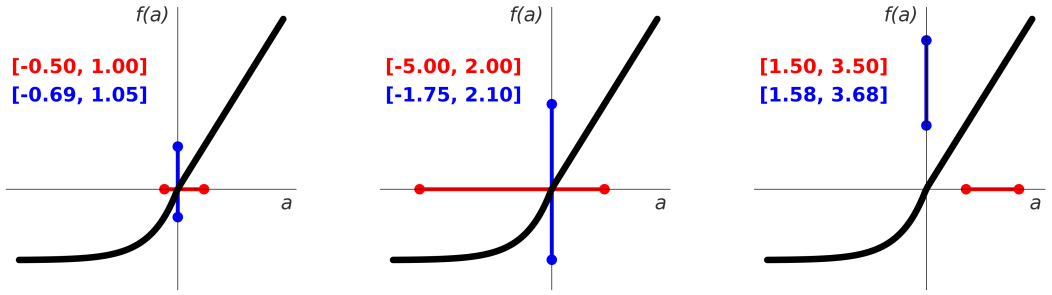
very deep networks may be trained effectively. Moreover, the residual block has also two other important advantages. Firstly, the output of the block already contains an identity (input to the block), therefore, the output does not have to be learned from scratch, but only a residuum to the input is learned which is an easier task [20]. Secondly, since there is a branch with identity, the network may decide on its own whether it is still advantageous to extract new features by the branch containing the layers or just pass the identity.

The internal covariate shift is addressed by normalization techniques. Specifically, a very popular block is Batch Normalization (BatchNorm) that performs a normalization to a learnable distribution for each feature [25] along the batch axis. A batch represents a whole set of training examples, however, BatchNorm can also be applied on mini-batches, i.e. a subset of training examples. Since the input to a hidden layer is always normalized by this block, the internal covariate shift is reduced, and bigger learning rates can be set resulting in quicker training. However, for small mini-batches BatchNorm is not appropriate as it uses statistical information derived from just a small number of samples for normalization. To deal with the internal covariate shift in case of small mini-batches, three main attitudes (LayerNorm, InstanceNorm, GroupNorm [60]) are presented in the literature performing the normalization along the feature (channel) axis instead of the batch axis. Also, instead of adding an extra normalizing block, SELU activation function [32] (eq. 2.1) with a so-called self-normalizing property[1] can be incorporated into the network. This property enables preserving approximately zero mean and unit variance throughout the network and thus mitigating the internal covariate shift [32] (Figure 2.4). For a proper working of SELU, standardized input data are necessary as well as LeCun initialization of the weights [32].

$$SELU(a) \approx 1.05070098 \cdot \begin{cases} a, & \text{if } a > 0. \\ 1.67326324 \cdot (e^a - 1), & \text{if } a <= 0. \end{cases} \tag{2.1}$$

Many extensions of gradient descent (vanilla learning algorithm) exist to prevent slow learning or getting trapped in local minima. In all cases, however, it is recommended to use sufficient mini-batch size to reduce stochasticity of movement to the global minimum. Commonly, a learning rate scheduler is used along with a chosen learning algorithm to perform coarse to fine learning. Today, the state

[1]SELU was designed according to Banach fixed theorem where the fixed point is zero mean and unit variance [32]. If the current point (i.e. current mean and variance of the activations from the given convolutional layer) is not very far from the fixed point (i.e. zero mean and unit variance), the current point is attracted to the fixed point. Even though SELU was introduced as a BatchNorm replacement in feed-forward neural networks [32], it was also used in CNNs [6, 10]



(a) SELU repairing too low variance and positive mean

(b) SELU repairing too high variance and negative mean

(c) weakness of SELU

Figure 2.4: SELU principle. Every time, SELU intends to transform the input (red) to the output (blue) with approximately zero mean and unit variance. In (a) we suppose low variance of the input that should be increased to the desired one (unit variance). This is ensured by SELU's design (eq. 2.1) containing regions with slope bigger than one enlarging the variance [32]. Similarly, in (b) we suppose too high variance of the input that should be decreased. Since SELU also contains regions with a slope smaller than one (mainly the saturation region), the output's variance is decreased [32]. These changes in variance through SELU also positively affect the output mean that is closer to zero in both cases (a, b). Importantly, SELU is able to preserve normalization, however, it does not normalize actively. Therefore, approximately standardized input is necessary, otherwise, SELU would not work as intended (c).

of the art learning algorithm is adaptive moment estimation (Adam) [31] that combines advantages of using momentum as well as using learning rate modification (depending on a gradient magnitude of each parameter). Even though Adam is a good first choice optimizer due to its fast convergence, in recent years it was investigated that it generalizes worse than stochastic gradient descent (SGD) with momentum [11, 30, 38, 59]. Regarding this fact, several extensions of Adam were published aiming to improve Adam's generalization performance, such as Padam [11] or AdamW [38].

From the foregoing it is evident that learning itself is not an elementary task. However, even after promising learning, the CNN model can be far from the expected performance. In biomedical area, a class imbalance problem is common forcing the CNN to classify each element of the grid-like data as the majority class (e.g. background) despite the fact that our object of interest is labeled as a minority class. Another problematic situation (overfitting) is when the model does not generalize well, i.e. it provides good results on the training set, but it struggles on the validation or testing set.

To address class imbalance problem, various loss functions were developed to improve the network's performance in learning the minority class. For example, weight-

ed cross entropy (or similarly balanced cross entropy) [1] can be calculated (instead of cross entropy) with a bigger weight on classifying the minority class properly. Also, to force the network to learn better the difficult examples (whose prediction was far from the ground truth), focal loss [1] can be used as the extension of balanced cross entropy. Popular loss functions are also these based on the overlap of the network prediction and the ground truth. For instance, Dice loss [1] is widely used or its more general form Tversky loss where false negatives and false positives may be weighted unequally. However, many other losses exist such as a combination of both balanced cross entropy and Dice loss (exponential logarithmic loss, combo loss [1]) or losses based on the distance between the segmentation prediction and ground truth (several alternatives of Hausdorff distance loss [1]). Also, dataset strategies exist to make the classes more balanced such as undersampling (i.e. random deletion of examples in the majority class) or oversampling (i.e. copy of examples in the minority class) [8].

As far as the overfitting is concerned, there are several strategies how to address it. A straightforward and very popular method (so-called early stopping) is to stop model learning in the case of stagnating or degrading of the validation results [19]. Another option is to regularize the absolute value (L1 norm) or squared value (L2 norm) of the weights by incorporating the penalization of the weights into the loss function. The idea behind the norm regularization is the fact that a model with bigger weights is able to produce more complex functions and therefore is prone to overfitting [19]. The third popular approach is training the network with Dropout [19] when each neuron has a certain probability of being excluded from the training process in the given iteration. In other words, in each iteration, a slightly different model is trained meaning that the network cannot fully rely on a given connection but rather is forced to learn more generally to perform well also in the case of the connection drop. If no strategy (including their combinations) eliminates overfitting, the model itself may be switched to a simpler one with a reduced capacity of learning. However, there is always a trade-off between the overfitting and underfitting (a situation when even training results are far from our expectation), therefore, the appropriate model for a given task must be chosen carefully. Lastly, overfitting can also be suppressed by a bigger training dataset that should describe a given problem more generally than the original one. This is done either by additional real data acquisition or synthetically by various augmentations.

2.3 CNNs for 3D image data segmentation

CNNs experience immense popularity as evidenced by a growing number of publications regarding the 3D image data segmentation using CNNs. Despite the huge

number of approaches, however, two main viewpoints can be distinguished based on the dimensionality of the utilized CNN. Either the 3D image data are divided into slices and processed by 2D CNN [17, 40] or directly 3D CNN [14, 27, 44, 63] is applied on the data which can be preceded by dividing the data into subvolumes (patches) [27, 63] due to GPU memory limits. Next, in several publications, the authors use more than one CNN model. For instance, the object of interest can be only roughly located using the first CNN, and subsequently a precise object segmentation is performed by the second CNN just in a limited area given by the first CNN prediction [5, 52]. Another example is merging the results of three 2D CNNs trained on different planes [2, 47]. Also, the combination of 2D and 3D CNNs may be seen in the literature [45, 61]. However, multi-network solutions are more difficult to train, therefore, a lot of publications aim to suggest only one CNN model with advanced architecture. In the following paragraphs some of the milestone CNN architectures for a semantic segmentation are described.

Fully Convolutional Network (FCN, Figure 2.5) is a revolutionary architecture transforming a classification CNN into the segmentation one [37]. As its name implies, it replaces fully connected layers (typically presented at the end of classification networks) by 1×1 convolutions making the network fully convolutional. This replacement is very advantageous as the input size to the final layers is not strictly defined by the number of neurons in fully connected layers [37]. Therefore, a bit larger input may be considered at the beginning of the final layers meaning that a class probabilistic image is obtained at the output of FCN rather than a class probabilistic vector. Since several max pooling layers are presented in FCN architecture, the image output of the final convolutional layer can be viewed as downsampled segmentation prediction. To obtain the segmentation in the original resolution, the authors tried several upsampling strategies (Figure 2.5) with the conclusion that FCN-8s performance is the best due to skip connections [37]. These were adopted in many following architectures as they add to the segmentation prediction from the feature-rich level also the finer details (due to bigger spatial resolution in the shallower levels) necessary for sharper borders of the final segmentation prediction.

FCN was followed by several architectures (e.g. DeconvNet [48], SegNet [4], U-Net [51] – Figure 2.6) trying to enhance the upsampling part. Specifically, so-called encoder-decoder architectures were published having the decoder part symmetric to the encoder. The deeper decoder enables to learn better upsampling resulting in the segmentation prediction improvement.

For biomedical image segmentation the U-Net architecture was a breakthrough as it enables training a precise model even with a limited dataset (common in biomedical tasks) [51]. Again, it uses long skip connections that make the segmentation prediction more accurate and also, from the training point of view, the skips reduce

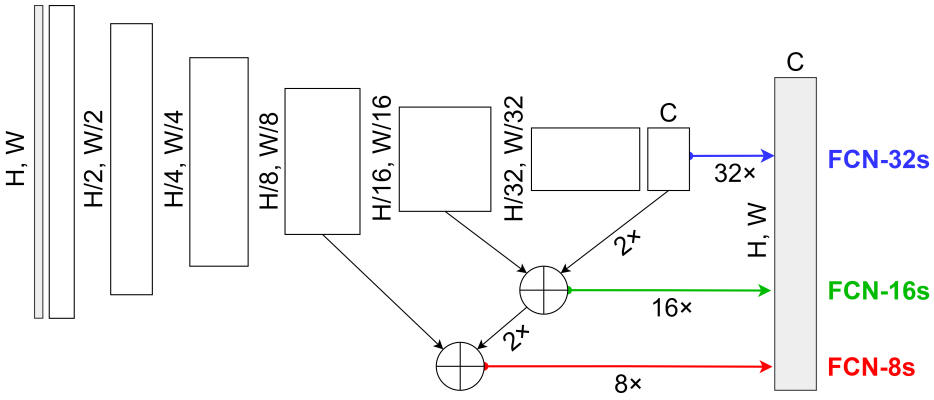


Figure 2.5: Fully convolutional network. The input with dimensions H, W is processed by the encoder outputing feature-rich activation maps further used for predicting probability of each class (C classes). Importantly, the output of the encoder is a probabilistic *image* with dimensions $H/32, W/32$, i.e. downsampled segmentation prediction. To obtain the prediction in the original resolution, three upsampling strategies were tried (FCN-32s, FCN-16s, FCN-8s). FCN-8s has the best performance as it uses the skip connections from two shallower levels.

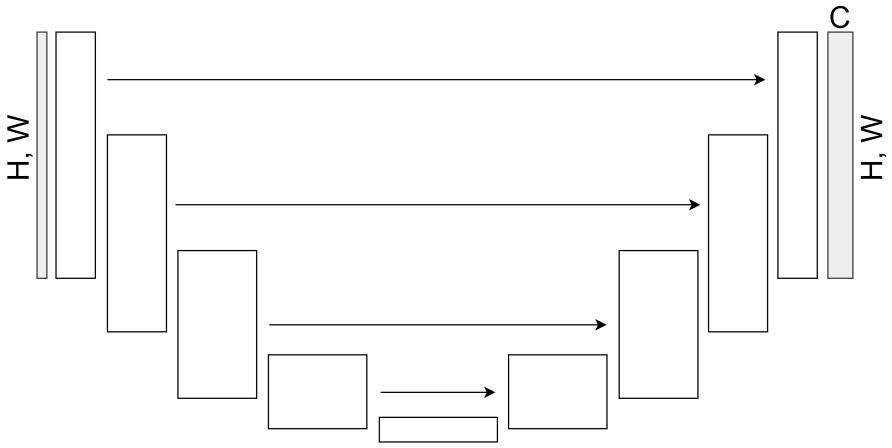


Figure 2.6: U-Net – encoder-decoder architecture. The input with dimensions H, W is processed by the encoder (left part) from which the feature-rich activation maps are upsampled back by the decoder (right part). Importantly, the decoder has a bigger capacity enabling better control over the upsampling. Also, the segmentation prediction (containing C classes) is improved by adding finer spatial details from shallower layers of the encoder using long skip connections.

the vanishing gradient problem. There are many extensions of U-Net in the literature [1]. For instance, several 3D CNN architectures inspired by U-Net were developed such as 3D U-Net [14] or V-Net [44]. Another example is fully convolutional DenseNet [26] that may be considered as U-Net (with fewer levels) with Dense blocks in the encoder as well as the decoder part. In Dense block, the input of each convolutional layer is forwarded to the inputs of *all* remaining convolutional layers

via skip connections to ensure maximal information flow [26]. Regarding raising popularity of attention mechanisms, Attention U-Net [49] was proposed having the attention gate in several levels of the U-shape architecture. The gate is able to learn where the object of interest roughly is located and based on that the feature activations are weighted to suppress the features from irrelevant regions and highlight the really important ones for the object [49]. U-Net++ [65] redesignes long skip connections to reduce the semantic gap between the encoder and decoder part. Specifically, U-Net++ utilizes nested dense convolutions to bridge the encoder and decoder part [65]. Furthermore, the model is deeply supervised using multiple outputs from the bridge with the highest resolution [65]. Probably the last extension of U-Net published to this day is U-Net 3+ [24] that implements so-called full-scale skip connections utilizing activation maps from multiple scales. Again, deep (multi-output) supervision is utilized, but in contrast to U-Net++, the outputs from several scales are used for learning [24].

Several architectures focus on enhancing the encoder part by adding multi-scale contextual information. For instance, DeepMedic [27] uses an architecture with two branches (encoders) each processing the input with different spatial resolution. In other words, local as well as global information is extracted resulting in more accurate segmentation prediction. Other architectures [64, 12] add the contextual information by utilizing specialized blocks based on pyramidal processing of already extracted features. For instance, a pyramid pooling module can be placed at the end of the encoder as proposed in Pyramid Scene Parsing Network (PSPNet) [64]. The pyramid pooling module uses parallelly four kernels with different sizes to perform information summarization (pooling) in different scales. Pooled feature maps are subsequently reduced in channels by 1×1 convolution, upsampled and concatenated to the original features (transferred from the input of the pyramid pooling module) resulting in the incorporation of the contextual information into the segmentation problem [64]. A similar approach, proposed in DeepLabV2 [12], is to extract the contextual information using four parallel dilated (atrous) convolutions with different rates (so-called Atrous Spatial Pyramid Pooling module, shortly ASPP). Generally, all versions of DeepLab are based on dilated convolutions since for segmentation purposes it is advantageous to preserve the spatial resolution of the feature activation map as much as possible (to avoid excessive upsampling resulting in fuzzy boundaries of the segmentation prediction) [12].

3 Proposed methodology

3.1 Available data and hardware

The available data (Table A.1) consist of 21 micro-CT datasets of mouse embryos in various developmental stages along with their binary masks defining the craniofacial cartilage (specifically the nasal capsule and the Meckel's cartilage as depicted in Figure 3.1).

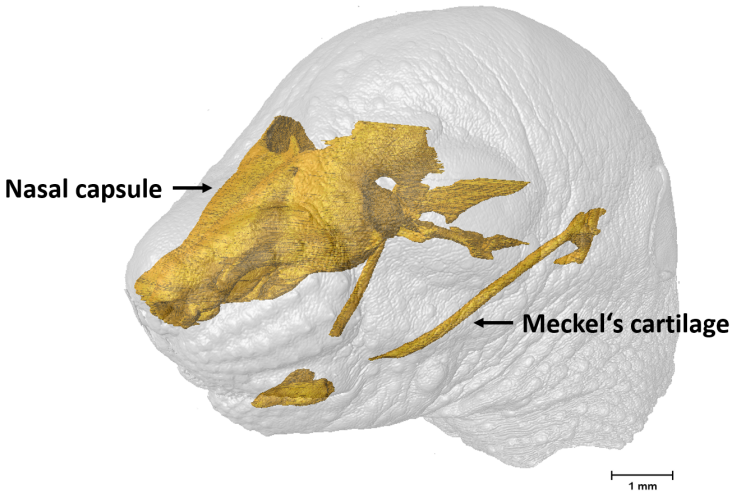


Figure 3.1: Craniofacial cartilage in the context of a head.

For simplicity, only the embryos from the 17.5-th day of development (E17.5) were used in the following experiments. Specifically, there were 14 E17.5 datasets consisting of 11 wild types and 3 mutants. The voxel size ranges from 4.5 µm to 6.2 µm depending on the dataset and the voxel itself was always isometric. The median size of the micro-CT data was $1348 \times 1916 \times 1419$ meaning a work with the memory-intensive data. Although the micro-CT data as well as the binary masks were already available as a stack of TIF images, it is necessary to get acquainted also with the origin of the data to fully understand them.

Before the micro-CT measurement, each embryo was stained according to protocol in [58] which may be divided into three main steps. Firstly, the sample was fixed by formaldehyde to prevent drastic changes in the shape (due to autolysis of the *ex vivo* sample). Next, for enhancing a penetration of the staining solution, the embryo was dehydrated in an alcohol series and then again rehydrated to get the original shape. Finally, to avoid the movement artifacts, the embryo was embedded to an agarose gel which keeps the sample hydrated and ensures its stable position. After the sample preparation, the data acquisition was performed using *GE Phoenix v‖tome‖x L 240* micro-CT with these settings: anode voltage 60 kV,

cathode current 200 µA, X-ray beam filtering using a berylium window of the X-ray tube and an additional 0.2 mm aluminium filter. The reconstructed 3D image data were then manually rotated in *VG Studio MAX* software so that slices are perpendicular to the mouse embryo. The reason for this is the fact that it is not guaranteed that the embryo was placed into the plastic tube with agarose exactly perpendicular to the X-ray tube during the measurement. Next, axial slices were exported as DICOM images that were used for a manual segmentation of the craniofacial cartilage. Specifically, the cartilage was manually segmented just in a chosen subset of axial slices subsequently used for deriving the segmentation in intermediate slices using a linear interpolation followed by binarization [33]. Finally, both the greyscale data and the binary mask were exported as a stack of TIF images that were used in the master thesis.

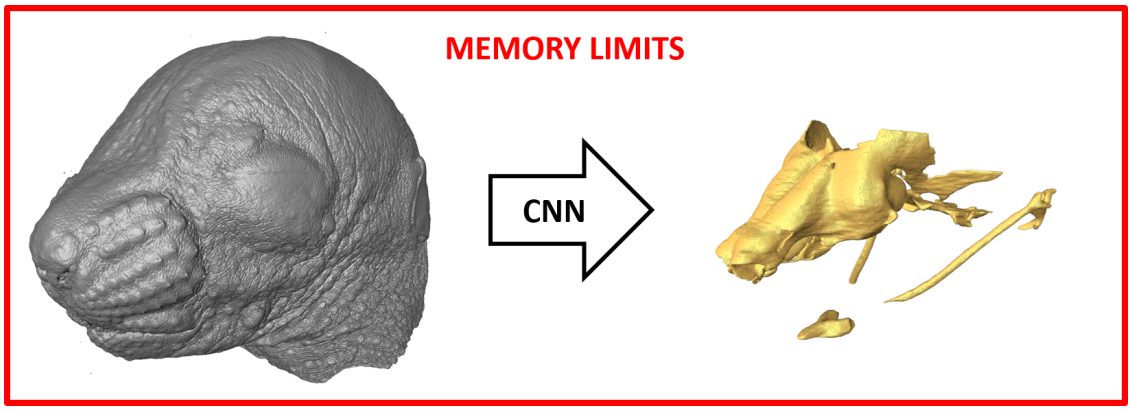
Regarding the hardware, a computer equipped with two separate *NVIDIA QUADRO RTX 5000* GPUs (each with dedicated GPU memory 16 GB), *Intel® Xeon® Gold 6154* processor working at 3.00 GHz and 255 GB RAM was available for experiments.

3.2 Pipeline

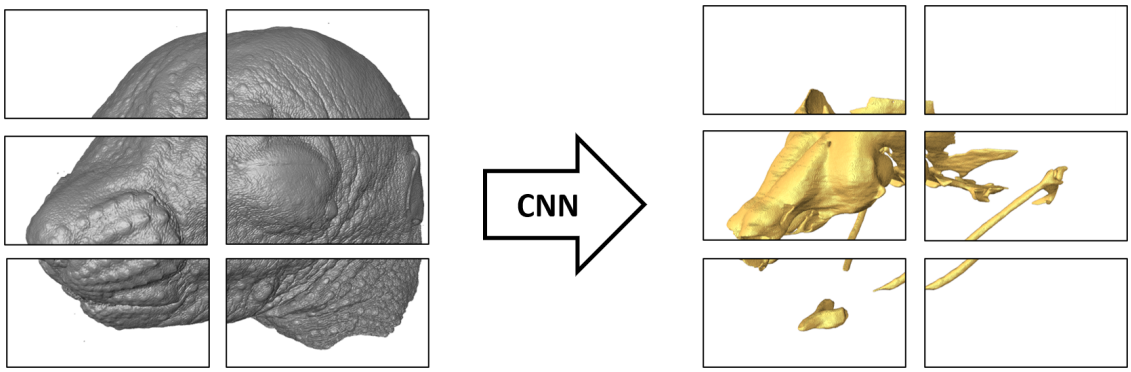
The selection of the pipeline for the given segmentation problem was affected by the available data and hardware (section 3.1) as well as by a previous work in this field [42]. In the previous research, U-Net performed a slice by slice segmentation meaning that the 3D information (naturally contained in micro-CT data) was neglected. Therefore, the master thesis focuses on incorporating the 3D information into the segmentation with a hope of getting even better results. Regrettably, the craniofacial cartilage is oblong in all directions meaning that even with a reasonable rate of downsampling the neural network cannot see the cartilage (or at least its sufficient part) at once because of memory limits. Nevertheless, the hypothesis is that even 3D *piecewise* segmentation (Figure 3.2) can enhance the segmentation results since the neural network can benefit from surrounding slices.

Before describing the pipeline in more detail, it should be stated that the binary ground truth mask was revisited to be more suitable for the 3D approach. As stated in section 3.1, the binary mask was manually segmented just in one plane meaning that discrepancies in the other planes were presented. These discrepancies would unnecessarily contribute to a noise component of the model's expected test error, therefore each binary mask was filtered by $5 \times 5 \times 5$ median filter[1] as proposed in [33]. The smoothing is not ideal as it totally omits the data information which may lead to inaccuracies. However, because these cases were rather minor,

[1]The median filtering was performed using ImageJ plugin *3D filters* [50].



(a) 3D segmentation



(b) 3D piecewise segmentation

Figure 3.2: The concept of 3D piecewise segmentation.

this smoothing approach seems to be sufficient for CNN training as we suppose that well-trained CNN should not overfit to the local inaccuracies. The binary mask before and after the smoothing can be seen in the appendix Figure B.1, the filtered mask was supposed to be the ground truth.

The whole pipeline of the 3D piecewise segmentation can be seen in Figure 3.3. Available data were already reported in section 3.1, the preprocessing and postprocessing phase will be described in the next paragraphs. The CNN processing part will be thoroughly described in chapter 4 and the evaluation in chapter 5.

3.2.1 Preprocessing

Regarding the 3D piecewise segmentation, the pieces (patches) of available data were supposed to be the input for the network. Nevertheless, the patches in the original resolution contained very local information insufficient for a good performance of the network. Therefore, adding more global information through downsampling the data (Figure 3.4(b)) was inevitable. Specifically, the original slices were four times downsampled (as in the previous research [42]) meaning that the network

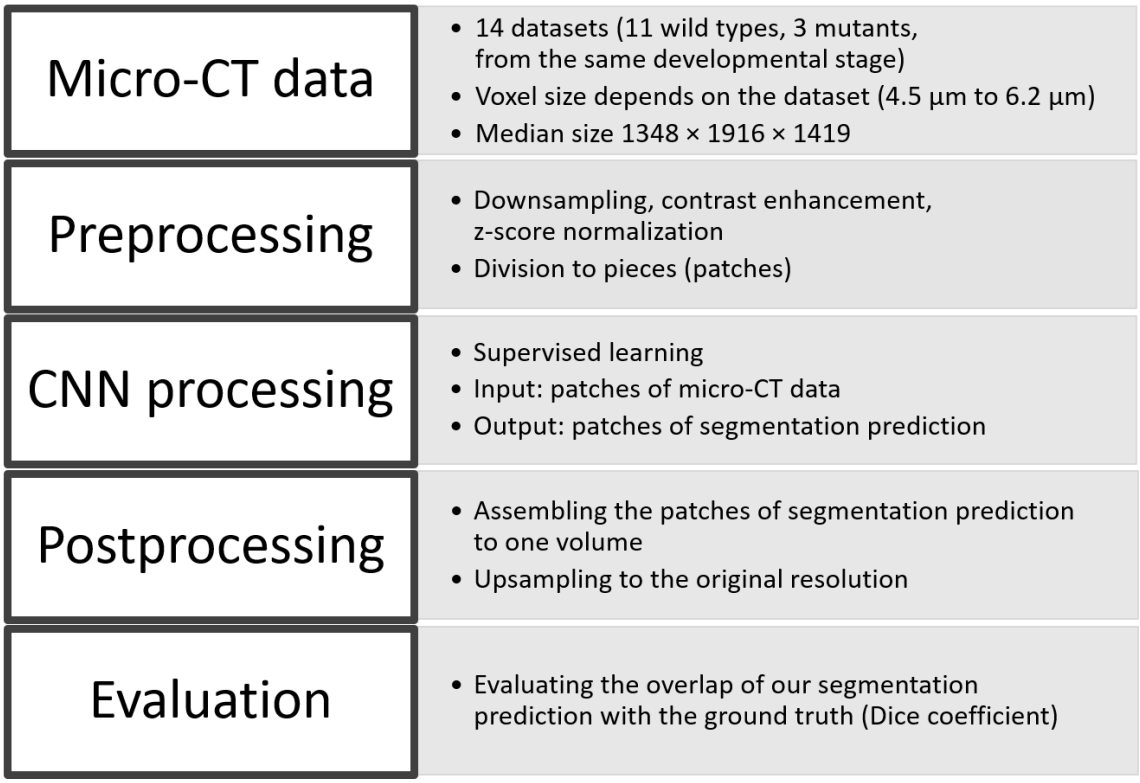


Figure 3.3: Pipeline of the 3D piecewise segmentation. The block scheme should be read from top to bottom.

could exploit both the global information in the xy plane as well as the local information using the surrounding slices. After the downsampling, the slices for each dataset were center cropped or padded to 320×448 size in order to unify the size of slices for different datasets [42]. Also, since both *uint16* and *uint8* data types were presented depending on the dataset, the data type was unified to *uint8*.

Typically, a histogram of micro-CT data (Figure 3.4(c)) contained three peaks – area outside the sample (peak A), agarose (peak B) and mouse embryo (peak C). Nevertheless, since the embryo had a high standard deviation of voxel intensities, the peak C overlapped with the B one. As we are interested in the embryo and its tissues, a window-level transformation (Figure 3.4(c)) was applied to the whole 3D image data to enhance the contrast between tissues. The contrast enhancement (Figure 3.4(d)) results in bigger differences between tissues and better learning of the network since the minor changes in parameters do not change the loss so drastically.

Generally, the network supposes centered and scaled data so that the activation functions really incorporate a nonlinearity into the process. Normalizing each patch separately may be inadequate because of the different data distribution in each patch. Moreover, normalizing the patch with a homogeneous area may lead to noise

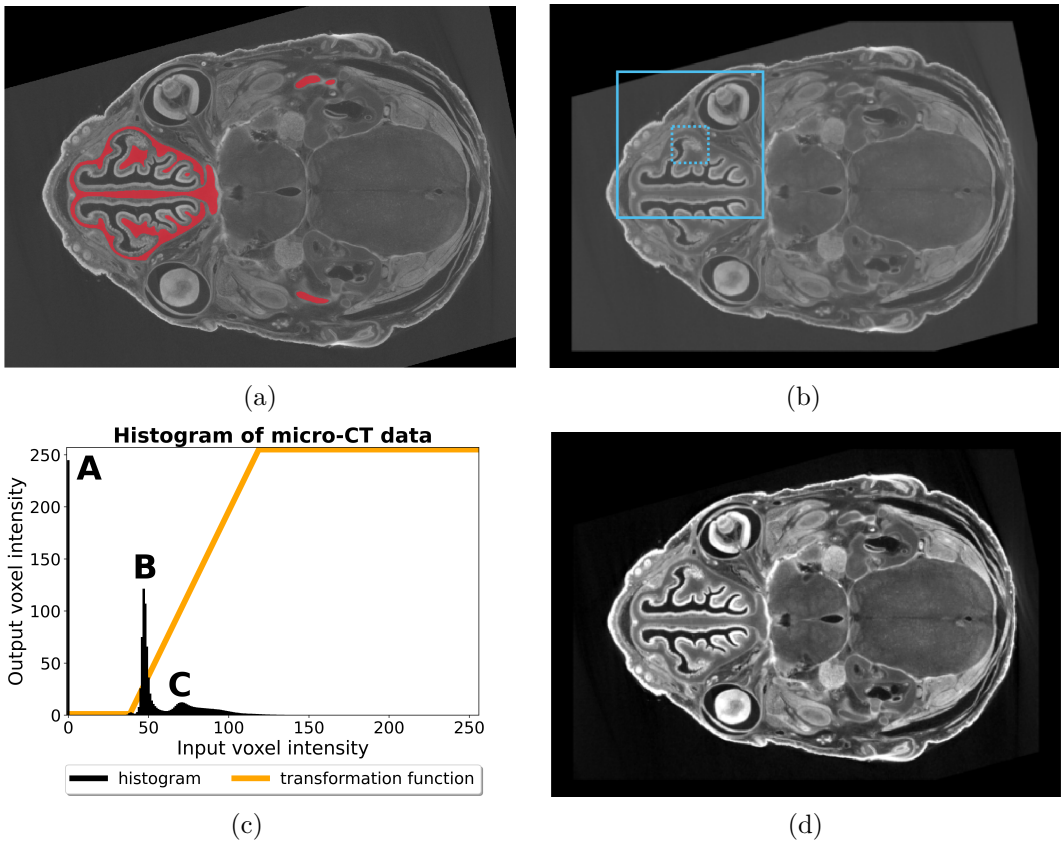


Figure 3.4: Preprocessing steps. Example of the original slice with the craniofacial cartilage colored by red (a), the same slice after the downsampling and padding to the 320×448 size (b) and after the contrast enhancement (d) according to the transformation function colored by orange (c). For each dataset, the transformation function was automatically derived from the histogram of micro-CT data also shown in (c) without bin counts. Additionally, in (b) it is demonstrated how downsampling enlarges the field of view (from dotted blue to solid blue) in the given patch.

increase. Therefore, the whole 3D image data were normalized by z-score before the division into the patches.

Assuming 3D piecewise segmentation, the preprocessed data had to be divided into the patches with size equal to the input size of the proposed CNN. Since the important structures could also be found exactly in the place of division, the patches were intentionally overlapped.

Supposing CNN processing of the preprocessed patches in the next step of the 3D piecewise segmentation pipeline, the decision about the way of feeding the network with patches had to be made. Firstly, the patches could be saved in extra files on a disk and afterwards loaded directly to the network (i.e. offline learning). Secondly, as a sufficient amount of RAM was available, all training embryo datasets might be loaded simultaneously into the RAM and subsequently preprocessed online implying a save of the disk space. However, even though that all embryo

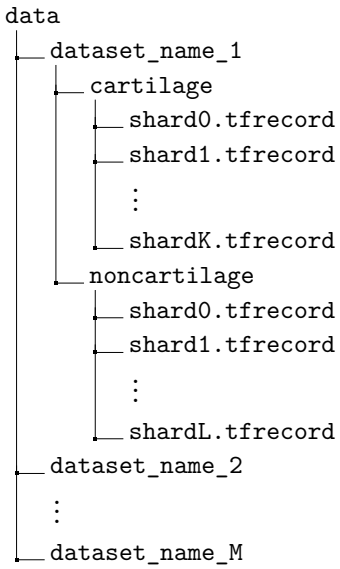


Figure 3.5: Directory tree used for saving patches. The patches were saved in multiple TFRecords (a binary storage format proposed for loading the data efficiently into the network using Tensorflow API). For better parallelization, the patches were divided and saved in multiple smaller files (shards) instead of one big file. Intentionally, the patches containing the craniofacial cartilage were saved in extra shards to deal better with the class imbalance problem in the CNN processing phase. According to the directory tree, disjunctive training and validation sets were ensured. For clarity, one shard contains several patches, and each patch contained two parts: micro-CT data and corresponding binary mask.

datasets were saved on SSD disk (quicker data access than HDD), loaded using multithreading approach (i.e. in parallel) and only representative slices[2] were used for preprocessing parameters calculation, still it took at least one minute for each dataset to be prepared for the training due to its size. Assuming various experiments (regarding network's learning) with several datasets, fully online learning was unacceptable. Nonetheless, fully offline learning was also limiting since we could not profit from a subsequent augmentation process so much[3]. Therefore, a compromise between offline and online learning was implemented. Specifically, patches with bigger size than the network's input size were saved on the disk (Figure 3.5) and then (during the learning) they were online augmented and cropped to the network's input size.

[2]Specifically, in each dimension three slices (distant to each other) were chosen and considered as the representative slices. The calculation of the preprocessing parameters using the representative slices was much faster and closely approximated the solution derived from the whole 3D image data.

[3]Imagine loading a given patch from the file and applying geometrical transform on it. The transformed patch will contain blank areas commonly filled by zero. Therefore, for the augmentation process it is more advantageous to know also the neighborhood of the given patch since then the blank areas can be filled by real voxel intensities.

3.2.2 Postprocessing

As the network performed piecewise segmentation, the pieces (patches) of predictions had to be put together in the postprocessing phase. Supposing algorithm taking care of dividing to patches, the exactly reverse algorithm was used for assembling them. Moreover, as the preprocessed patches overlapped, their probabilistic segmentation predictions were merged and after that rounded to obtain the binary mask instead of the probabilistic mask. Finally, the binary mask was interpolated by the nearest neighbor to the original resolution since the 3D image data were downsampled in the preprocessing phase.

Merging of the segmentation predictions was performed by weighted arithmetic averaging (schematically depicted in Figure 3.6). Specifically, the prediction for each patch was multiplied by a weight mask having the bigger weights in the center of the patch. In other words, on the edges of the patch the prediction was supposed to be less accurate due to the lack of neighbouring information. However, if more overlapping patches agreed on the prediction in their edge part, they could outvote the patch giving the center prediction.

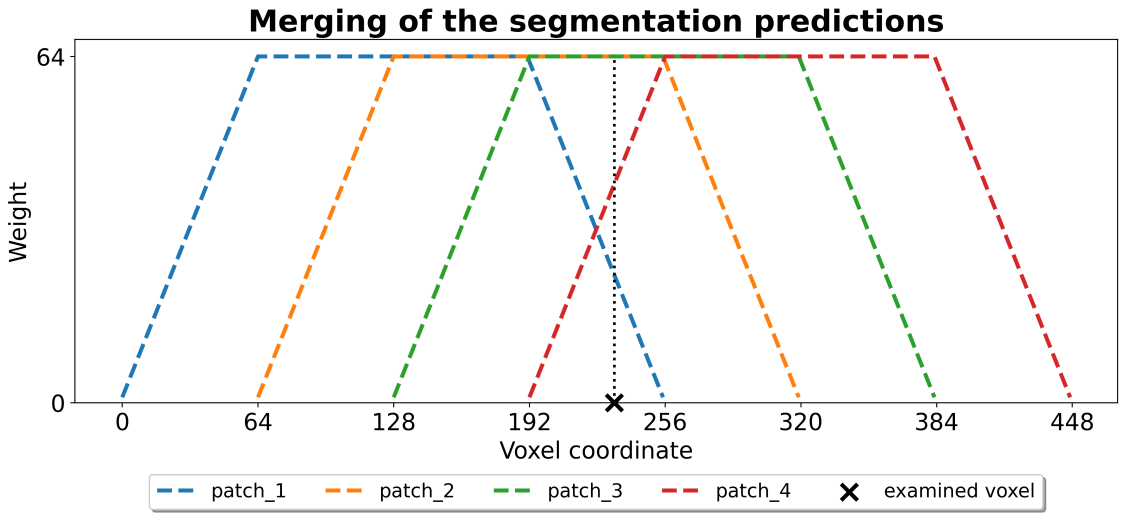


Figure 3.6: Merging of the segmentation predictions (schematically depicted in 1D). Consider a patch size equal to 256 and a stride 64 in the given dimension. The patches overlayed, thus, for an examined voxel, several segmentation predictions existed. The binary class for the examined voxel was derived as a weighted average of these probabilistic predictions followed by rounding. Note: The weight of the patche's prediction depended on the distance of the examined voxel from the center of the patch. The biggest weight (empirically set to a quarter of the patch size) was set in the center of the patch and its neighborhood. Then, the weight linearly decreased towards the edges of the patch and naturally beyond the edges the weight was zero.

4 Baseline network and its optimization

The master thesis focuses on the incorporation of the 3D information into the segmentation process. Although 2D CNNs can accept 3D inputs (set of slices), they cannot provide spatial information about the extracted features in contrast to 3D CNNs. Therefore, a 3D CNN solution based on V-Net [44] was proposed as a baseline network which was further refined in a subsequent optimization process.

4.1 Baseline architecture

V-Net is an encoder-decoder 3D CNN architecture depicted in Figure 4.1. There are five levels of multiple scales in V-Net from which the features are extracted through a set of $5 \times 5 \times 5$ convolutions followed by PReLU nonlinearity. The downsampling is done using strided convolution instead of pooling which may be viewed as learned information summarization. For upsampling, strided transposed convolutions are used making also the upsampling learnable. To achieve a better flow of the information, long skip connections are presented between the encoder and decoder part. Also, in each level, short skip connection is implemented through a residual connection ensuring better learning. Finally, the probabilistic prediction for each class is obtained using softmax activation function at the end of the network. [44]

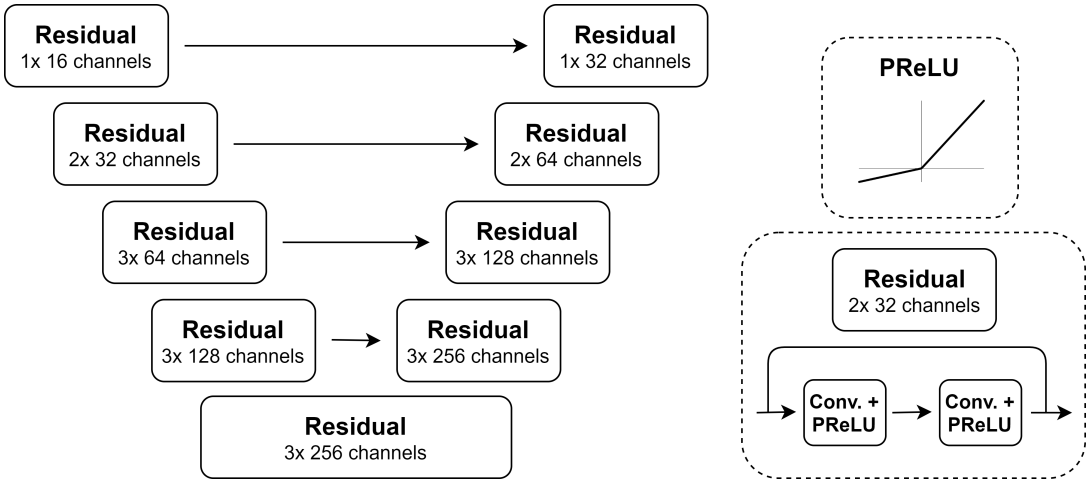


Figure 4.1: V-Net architecture (left) along with the blocks explanation (right).

The V-Net was further modified to be more suitable for the given segmentation problem:

1) V-Net's input size is designed for small compact objects of interest[1] which is not a case of the craniofacial cartilage. Thus, the input size was changed from $128 \times 128 \times 64$ to $256 \times 256 \times 24$ leading to major improvements in the segmentation of Meckel's cartilage (Figure 4.2). Naturally, enough global information in each plane (thus the input size e.g. $256 \times 256 \times 256$) would boost the segmentation performance even more, however, this was not possible because of memory limits. Assuming the input size $256 \times 256 \times 24$, the 5th level was omitted from the architecture since the depth size in the 4th level was only 3 (Figure 4.3).

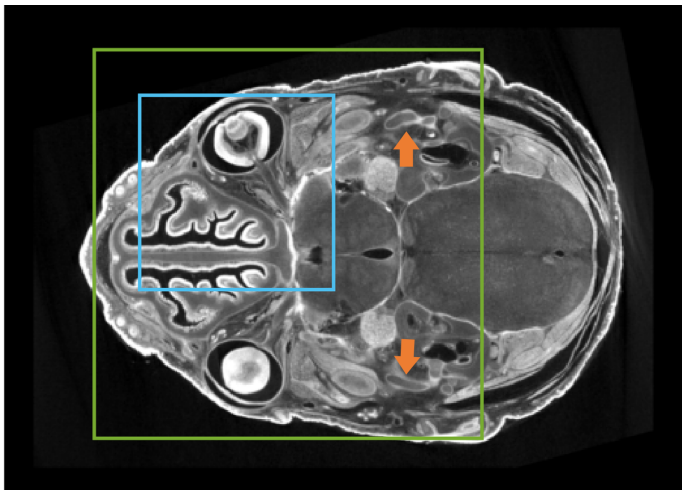
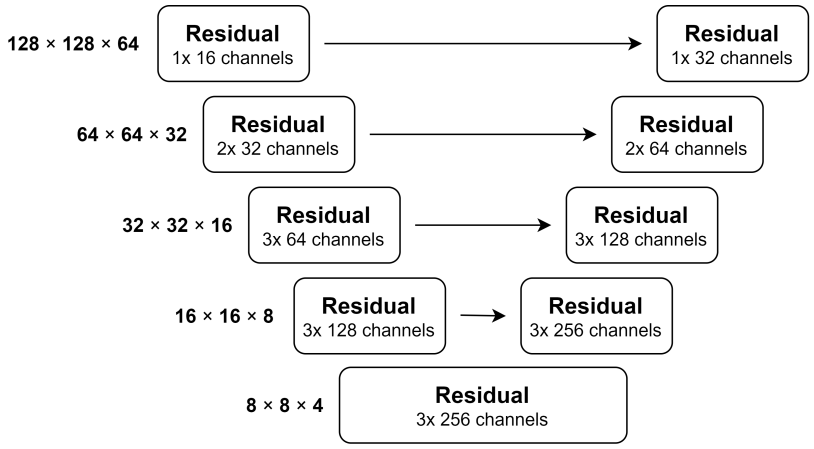


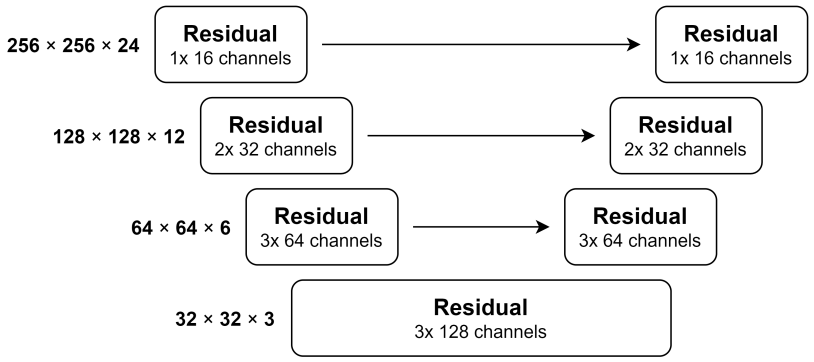
Figure 4.2: Change in input size (blue to green) helped to segment problematic parts (orange arrows) more properly. Note: the slice is after contrast enhancement.

2) Supposing only two prediction classes (craniofacial cartilage versus background), the softmax activation function was replaced by the sigmoid one for simplicity. In other words, the output contained only one channel representing a probability that a given voxel belongs to the craniofacial cartilage.
3) Assuming patches with very different contents, it is crucial to have sufficient mini-batch size for effective learning. In V-net the decoder has twice as large capacity than the encoder which is rather unusual in the literature and the authors did not provide any special meaning for that. Regarding this fact, the decoder capacity was experimentally reduced to the encoder one, which did not lead to a decrease in training results. Therefore, the architecture was adjusted to have the same capacity in the encoder and the decoder part (Figure 4.3) implying a bigger possible mini-batch size and also a smaller chance of overfitting.

[1]Specifically, the authors of V-Net dealt with a prostate segmentation from MRI volumes (PROMISE 2012 dataset) [44].



(a) V-Net



(b) Baseline model

Figure 4.3: Transformation of V-Net (a) into the baseline architecture (b).

4.2 Training the model

For the definition of the baseline model along with its training procedure, *Tensorflow 2* was used as one of the most popular frameworks for developing deep learning applications. The model was trained on GPU since it performs the matrix multiplications (frequently used in neural network training) much faster than CPU. Specifically, GPU parallelization was used as the computer was equipped with two GPUs. However, these GPUs were separate meaning that the network's memory footprint had to be lower than the dedicated memory of a single GPU. The parallelization was based on a mirroring strategy meaning that two identical networks ran on the two GPUs each processing one half of the mini-batch but updating the parameters according to the aggregated gradients from the two GPUs [56].

Before the network's learning, however, the loading pipeline (Figure 4.4) had to be implemented to get the preprocessed patches from shards into the network. For each class a separate data flow was created and step by step filled with patches

saved in the shards. To ensure sufficient shuffling of the training data, the shards as well as the loaded patches were shuffled in the given data flow. Next, the main data flow was created by sampling from the two separate data flows. Specifically, to prevent the class imbalance problem, the samples were taken from the cartilage flow with a bigger probability than from the noncartilage flow. As stated in subsection 3.2.1, the patches were intentionally saved with a bigger size than the network's input size to perform the online augmentation more efficiently. Thus, the next step of the loading pipeline was to create the augmented patches with the network's input size as described in the following paragraphs. Finally, the patches were again shuffled and placed into the mini-batch further used for the network's learning.

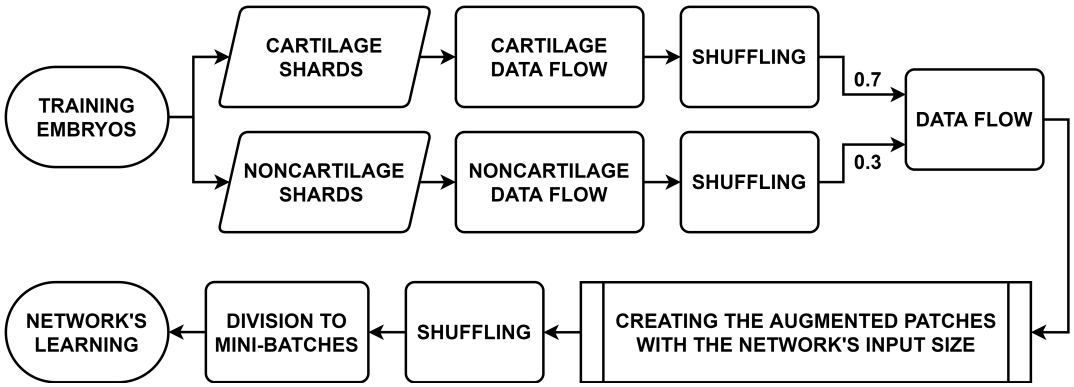


Figure 4.4: Loading pipeline. Note: For reproducibility, random seeds were set for shuffling as well as the augmentation process.

The augmentation process consisted of random flipping, shifting, zooming and noise addition applied with heuristically predefined probabilities (Table 4.1). Flips were seen even in the available (limited) dataset meaning that these transformations would probably be common also in unavailable (future) datasets and should have the biggest probability. Shifts simulated dividing the image data into the patches with another starting point of division. In other words, some of the structures in the original patch did not have to be presented in the shifted patch implying the regularization effect. Zooming in various axes imitated possible shape changes in the craniofacial cartilage in the case of mutant embryos. Additionally, zooming might be considered as changing voxel size artificially, making the network again more general. As stated in section 3.1, micro-CT datasets were manually rotated so that the slices are perpendicular to the mouse embryo as much as possible. Therefore, the rotation was omitted from the augmentation process. Also, no contrast augmentations were implemented as the contrast enhancement was already done in the preprocessing phase for all embryos. To prevent overfitting, noise augmentation was also added as a regularizer, even though the 3D image data itself had a high signal to noise ratio (due to long measurements).

Unfortunately, as far as the master thesis is aware, nowadays *Tensorflow 2* does not provide any application programming interface (API) for 3D geometric transformations. It is possible to call an extra Python function (*tf.py_function* [57]) implementing the 3D geometric transformations using the external libraries (e.g. SciPy), however, it requires a so-called eager execution instead of a graph execution [57] implying a remarkable slowdown of the loading pipeline. Therefore, for speeding up the pipeline, the geometric transformations were not applied in 3D but rather parallely on the stack of 2D axial slices. More precisely, a patch with size $256 \times 256 \times 24$ (the network's input size) was cropped from the bigger patch according to the geometric transformation parameters (e.g. shifting implies a shift of a cropping window). Finally, the cropped patch could be augmented using noise addition and the subprocess of creating the augmented patches with the network's input size was finished.

Table 4.1: Augmentation parameters. Note: geometric transformation had the same possible range of parameters in both dimensions of the slice.

Augmentation	**Probability**	**Additional information**
Flip	0.30	–
Shift	0.20	shift $\in [-32.0, 32.0]$
Zoom	0.15	zoom $\in [0.75, 1.25]$
Noise addition	0.10	noise $\sim \mathcal{N}(0, 1)$

Before the network's learning, several hyperparameters had to be defined. Unfortunately, available 14 datasets are not enough to create representative training, validation and testing sets. Therefore, tuning hyperparameters according to the validation set was not implemented and the hyperparameters were set heuristically. The number of epochs was set to 5 as more epochs did not lead to a considerable increase in training results. In other words, an overfitting area was supposed when the training results just slightly increased (less than 2 %) and the training was stopped. In each epoch, the whole training dataset was used for learning. The mini-batch size was 8.

Training with class imbalanced data can lead to predictions with high precision but low recall[2] [53]. This issue is addressed by Tversky loss [53], which was chosen as the loss function for the network's learning. The Tversky loss can be expressed by eq. 4.1, where $\vec{g}$ is a vectorized ground truth, $\vec{p}$ is a vectorized probabilistic

[2]Precision (also known as positive predictive value) represents a percentage of true positives (TP) in the set of all positives labelled by a model. Smaller the precision, the higher the number of false positives (FP). Recall (also known as sensitivity) represents a percentage of TP in the set of all positives given by the ground truth. Smaller the recall, the higher the number of false negatives (FN).

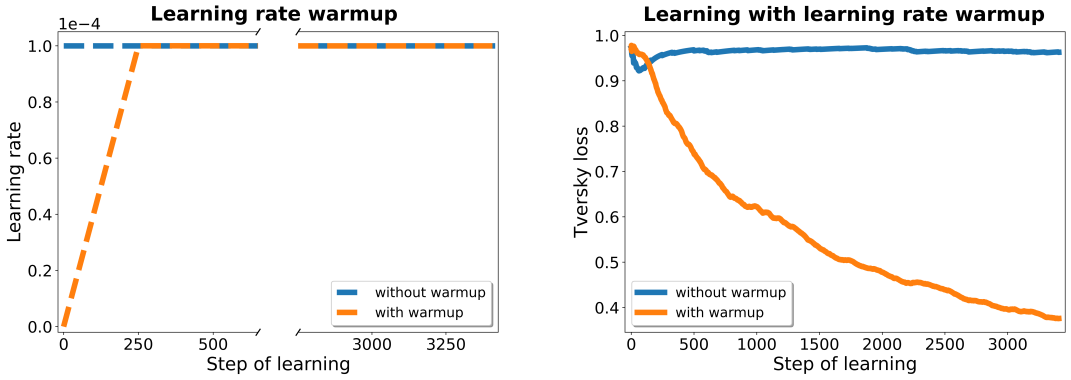
prediction of the network, β is a weight for false negatives and was empirically set to 0.7 and ϵ guarantees numerical stability.

$$TverskyLoss(\vec{g}, \vec{p}) = 1 - \frac{\vec{g} \cdot \vec{p} + \epsilon}{\vec{g} \cdot \vec{p} + \beta \cdot \vec{g} \cdot (1 - \vec{p}) + (1 - \beta) \cdot (1 - \vec{g}) \cdot \vec{p} + \epsilon} \tag{4.1}$$

AdamW [38] was chosen as an optimizer with an empirically set initial learning rate 1e−4 and weight decay 1e−6. As stated in section 2.2, AdamW is an extension of widely used Adam addressing its generalization problems discovered in the recent years. In AdamW paper, the authors point out that even though the weight decay can be considered as L2 regularization (commonly implemented in deep learning frameworks) in the case of stochastic gradient descent (SGD), this does not apply to adaptive learning rate optimizers such as Adam [38]. Using L2 regularization in Adam results in less penalizing the norm of parameters with bigger gradient magnitudes implying poorer generalization ability [38] (chapter C). For that reason, the Adam optimizer was replaced by AdamW that penalizes the norm of all parameters equally [38].

The initial learning rate 1e−4 worked well on non-augmented inputs, however, in the case of the augmented (more difficult) inputs, the learning often failed at the very beginning of training. This early-stage divergence phenomenon was identified in the literature as a consequence of undesirably high variance of the adaptive learning rate at the beginning of training [35]. Specifically, due to the lack of samples at the early stage, the estimation of the exponential moving average (used for the adaptive learning rate computation) is not as accurate as intended (i.e. the estimations have a high variance) leading to possible inappropriate movements in the optimization landscape and trapping in a bad local optimum in the worst case [35]. Several strategies can be used for mitigating this problem including learning rate warmup [35] depicted in Figure 4.5. Specifically, the learning rate was linearly increased from zero to the required initial learning rate for a few initial steps implying a more careful movement in the optimization landscape as well as avoiding the divergence at the beginning of training. To model coarse-to-fine behaviour of the learning, the initial learning rate was multiplied by $e^{-0.15 \cdot EpochNumber}$ each epoch.

During the training, three metrics were monitored: loss, accuracy and Dice coefficient (further used for the final evaluation). High accuracy and low Dice corresponds to a class imbalance problem, whereas low accuracy and Dice are a sign of training problems described in section 2.2. Therefore, it was necessary to monitor both metrics in each experiment. The metrics were treated with caution since they were related only to the patches and not to the whole embryo.



(a) (b)

Figure 4.5: The effect of learning rate warmup (a) on the learning (b). The warmup period was empirically set to initial 250 batches (a). It is evident that the early stage divergence phenomenon was mitigated using the warmup, although the learning was slowed down a bit at its beginning (b). Note: the learning curves in (b) were smoothed and represent the first epoch.

4.3 Optimization

The baseline model was further refined in a subsequent optimization process. Specifically, the master thesis emphasizes the utilization of advanced architectural modifications (presented in the literature in recent years) as they can greatly improve the segmentation performance compared to the vanilla architecture.

Atrous spatial pyramid pooling In order to compensate the removal of the 5th level from V-Net (Figure 4.3) and to add even more multi-scale information, an atrous spatial pyramid pooling (ASPP) module [12] was placed in the lowest level of the baseline architecture. Nonetheless, it would be naive to reuse the published ASPP without any change as the suitable dilation rates for convolutions should be set according to the input feature map size[3]. Thus, ASPP suitable for the baseline architecture was proposed (Figure 4.6) including $3\times3\times3$ dilated convolutions extracting features from various scales. No dilation was used for z axis since the depth size was only 3 at that level. Also, inspired by a newer version of DeepLab (DeepLabV3 [13]), $1 \times 1 \times 1$ convolution (summarizing the channel information for each voxel) was added to the proposed ASPP.

[3]From Figure 4.3(b) it can be derived that the input feature map size to ASPP was defined as $32 \times 32 \times 3$. In DeepLabV2 [12] e.g. a dilation rate 24 was used for the 2D input feature map meaning that in the xy plane it would cover the area 49×49. In other words, using the dilation rate 24 for our case would result in extracting features outside our feature map which is possible (by zero-padding) but does not make sense.

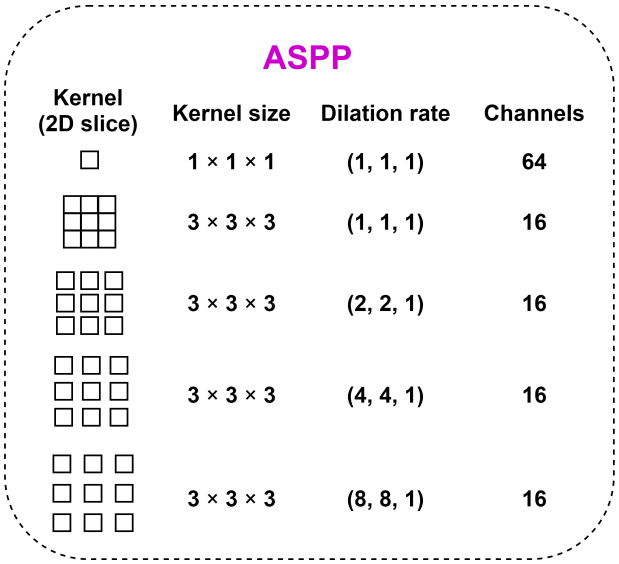


Figure 4.6: Proposed atrous spatial pyramid pooling (ASPP) module. The output of the module was derived using a concatenation of all channels (i.e. 128 channels).

SELU activation function Neither in V-Net nor in the baseline model the internal covariate shift (section 2.2) was considered. To address this problem, SELU activation function [32] described in section 2.1 and section 2.2 was chosen as the right option for mitigating the problem. A great advantage of SELU over the other normalization techniques is no need for hyperparameter tuning as well as no dependency on the mini-batch size.

Multi-output supervision To ensure more stable learning and high quality of the features in the network's whole depth, the learning was dependent on the Tversky loss in all levels of the CNN architecture. There are several implementations of the multi-output supervision (also known as a deep supervision) in the literature (e.g. [17, 24, 63]). Either the segmentation prediction of each decoder stage is upsampled to the original resolution and compared with the ground truth [24, 63], or the ground truth is pyramidally downsampled to match the resolutions in various levels of the decoder [17]. The second option was chosen in the master thesis. Specifically, the output of each decoder level was convolved by $1 \times 1 \times 1$ kernel followed by a sigmoid function yielding the segmentation prediction for the given level. Next, for each level, the Tversky loss was computed according to the match of the segmentation prediction and appropriately downsampled ground truth. Finally, to compensate various sizes of predictions in each level the overall loss was computed as the *weighted* average of the individual losses. More precisely, the weighted aver-

age was refined as the weighted sum, where the sum of the weights was equal to one (eq. 4.2).

$$w_l = \frac{\log\left(H_l \times W_l \times D_l\right)}{\sum_{k=1}^{L} \log\left(H_k \times W_k \times D_k\right)} \tag{4.2}$$

$H_l \times W_l \times D_l$ is the size of the segmentation prediction in a given level l, L represents the number of CNN levels and w_l is the weight for the individual loss derived from the level l. For clarity, the logarithm in the formula is necessary to alleviate the differences between the prediction sizes and to truly take advantage of the multi-output supervision[4] depicted in Figure 4.7. Moreover, the proposed eq. 4.2 can serve as a good starting point for further research since it is generally designed and thus reusable.

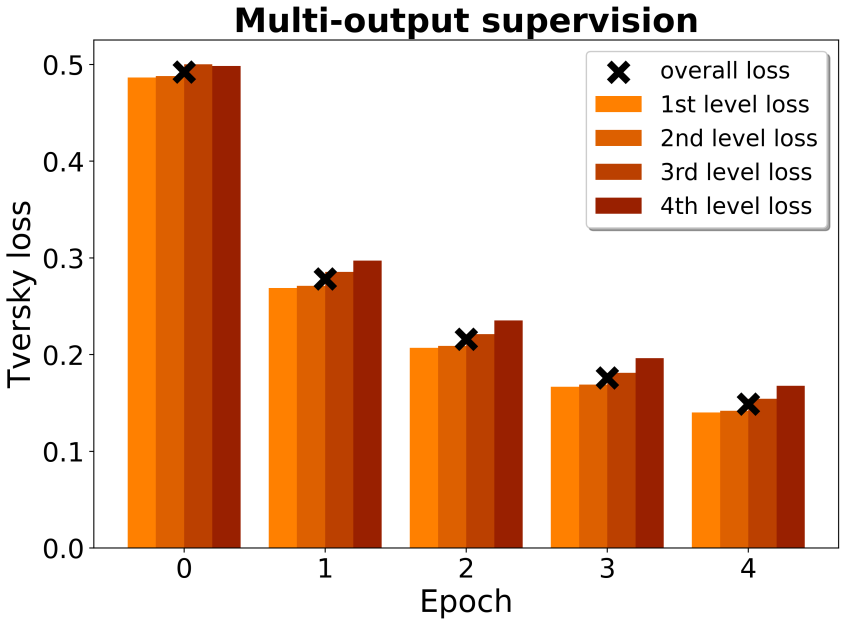
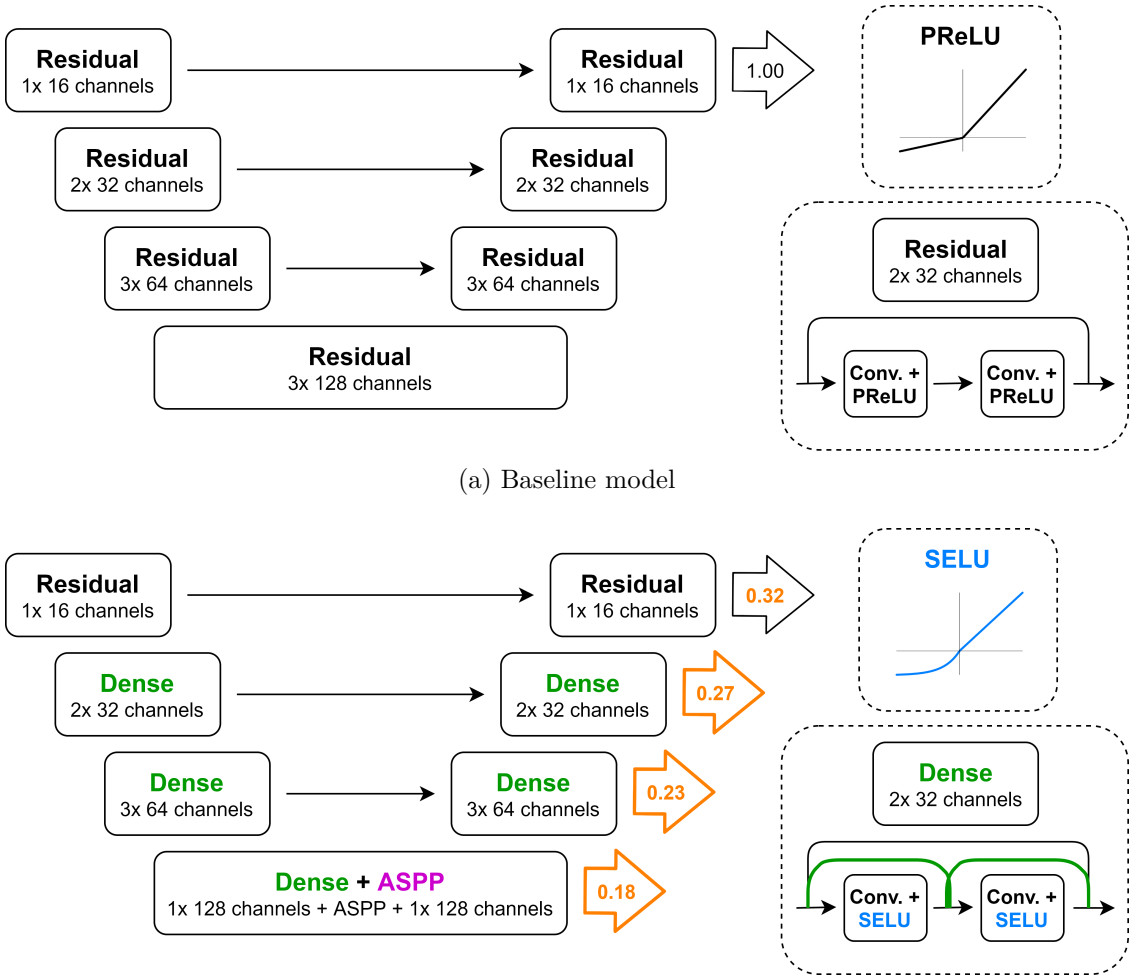


Figure 4.7: Multi-output supervision. From the individual losses after each epoch it is evident that the multi-output supervision ensured a high quality of the features in the network's whole depth.

Dense blocks In each CNN architecture level with series of convolutions, dense connections [26] were added to reuse the extracted features as much as possible. Because of the limited GPU memory, an additive connection was used instead of concatenation.

[4]Reminder from Figure 4.3(b) that the size of the segmentation prediction from the lowest level was defined as $32 \times 32 \times 3$. Using eq. 4.2, the weight for the loss from this level was 0.18. If the eq. 4.2 did not contain logarithms, the weight for the loss from this level would be 0.00 (after rounding to two decimal places).

The whole optimization process is schematically depicted in Figure 4.8 and evaluated in section 5.1.



(a) Baseline model

(b) Optimized model

Figure 4.8: Optimization process. Specifically, the atrous spatial pyramid pooling (ASPP) module (magenta), SELU activation function (blue), multi-output supervision (orange) and Dense blocks (green) were incorporated into the baseline architecture during the optimization process.

5 Evaluation & discussion

Since the available dataset was limited, 7-fold crossvalidation was performed to obtain more relevant results of the given experiment. Dice coefficient (eq. 5.1), commonly used for representing an overlap between a ground truth G and a segmentation prediction P [1], was chosen as the evaluation metric. To obtain comparable results, random seeds were set for random shuffling, augmentation operations and weights initialization. Every time, only one aspect in the experiment was changed to clearly see the effect of this aspect.

$$Dice = \frac{2 \cdot |P \cap G|}{|P| + |G|} = \frac{2 \cdot TP}{2 \cdot TP + FP + FN} \tag{5.1}$$

5.1 Optimization evaluation

First of all, the optimization process was evaluated using the available dataset of E17.5 mouse embryos. From Figure 5.1 it is evident that the optimization process gradually raised the median of Dice coefficient from 69.74 % to 80.01 % and also remarkably reduced the interquartile range of the results. The two outliers in Figure 5.1 represented an overstained embryo and a very different-looking mutant. Naturally, as these two samples exceedingly differed from the training set, the network's performance on these two samples was the worst. Therefore, having a representative dataset is equally important. The Figure 5.1 without the outliers can be seen in Figure D.1 in the appendix.

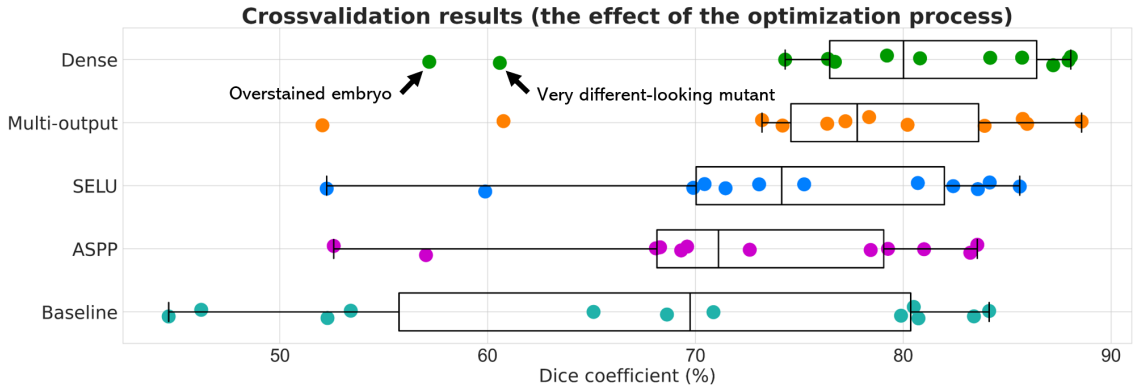


Figure 5.1: 7-fold crossvalidation results mapping the optimization process. The components on the y axis were added to the architecture in the previous step (e.g. SELU boxplot was created using baseline architecture with ASPP and SELU). The points represent particular embryos from which the boxplot for each experiment was derived. The box extends from the first quartile Q_1 to the third quartile Q_3, thus, its length represents interquartile range ($IQR = Q_3 - Q_1$). The length of whiskers is $1.5 \cdot IQR$. For clarity, the line inside the boxplots represents the median of Dice coefficient.

5.2 Effect of the preprocessing and augmentation

Several experiments were performed to justify some of the non-architectural steps of the proposed methodology (Figure 5.2). For all experiments, the optimized model (Figure 4.8(b)) was used. First of all, from the top row of Figure 5.2 it is evident that even though several regularization strategies were implemented, the network's performance still suffered from overfitting. Thus, reducing the capacity of the decoder to the encoder one as described in section 4.1 was an appropriate step since otherwise the overfitting would be probably even bigger.

Moving to the non-architectural experiments, the downsampling of the axial slices instead of downsampling the whole volume in the preprocessing phase was justified using the second top row of Figure 5.2. By downsampling also the z axis, additional global information was incorporated into the segmentation process which was viewed as the possible advantage for the network. On the other hand, the local information (i.e. details) about the cartilage was lost having a negative impact on the results exceeding the mentioned advantage. Another weakness of the downsampled depth scenario is the fact that in the postprocessing phase the upsampling had to be performed in three axes instead of two.

The next experiment in Figure 5.2 emphasizes the importance of the contrast enhancement in the preprocessing phase. From the training results it is evident that the network's learning was more challenging without the contrast enhancement as the network should identify the minor difference between the craniofacial cartilage and the surrounding tissues. The contrast enhancement was done using a simple window-level transformation, however, more sophisticated transformations can be exploited in the future. For instance, the input values in the window could be transformed to the output values using a nonlinear transformation function designed to enhance the cartilage even more. Also, the histogram equalization could be performed in the window to unify the histogram distributions between available micro-CT datasets.

Finally, when excluding the augmentation process from the training of the optimized model, the overfitting increased as it is depicted in the bottom row of Figure 5.2. Therefore, it can be stated that the hyperparameters of the augmentation were appropriately set, even though there is a great room for improvement. Both probabilities as well as ranges defined in Table 4.1 can be tuned, and also additional transformations can be exploited. Moreover, in the master thesis, only the global transformations were applied, however, the local transformations might also be advantageous since they can imitate local protuberances due to dysmorphogenesis of the embryo. For instance, it could be realized using a deformation

field with a few radial basis function peaks randomly placed into the field during the augmentation process resulting in local transformations at these places.

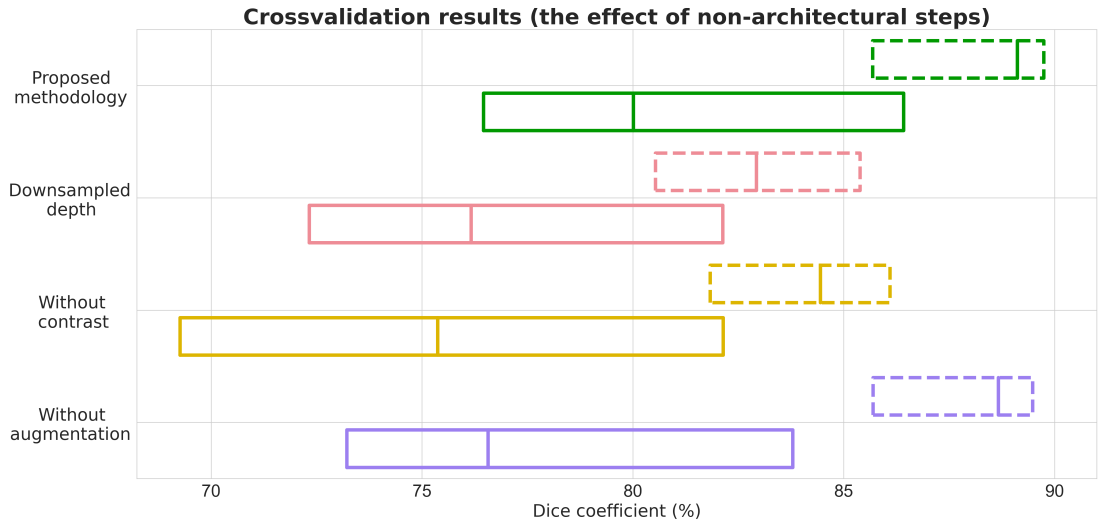


Figure 5.2: 7-fold crossvalidation results justifying the proposed methodology (see the text). The boxplots for each experiment (distinguished by color) were derived in the same way as in Figure 5.1. The solid boxplot represents the results on the validation set, whereas the dashed boxplot reflects the training results. For simplicity, whiskers and outliers were not drawn.

5.3 Evaluation using the extended dataset

The overfitting can also be addressed with a bigger training dataset. Therefore, in another experiment (Figure 5.3), also the other developmental stages (i.e. not only E17.5, see Table A.1) were incorporated into the learning process. Again, the 7-fold crossvalidation was used for the evaluation (Figure 5.3). It was assumed that the validation results of E17.5 embryos should increase in comparison with the previous experiments because of a more robust training set and thus less overfitting. Unfortunately, from Figure 5.3 it is evident that although the overfitting (i.e. the difference between training and testing results) was reduced, the mentioned assumption did not apply in general as the medians of Dice coefficients were comparable. Nonetheless, at least the interquartile range of the results was decreased using the extended dataset making the estimation of the network's performance more accurate.

Also, in Figure 5.4 the validation results for the non-E17.5 embryos are depicted. It can be seen that E15.5 embryos (e.g. Figure 5.5(b)) had generally the worst results since they were the least similar to E17.5 embryos (e.g. Figure 5.5(a)) making a majority of the training dataset (14 E17.5 versus 3 E15.5, 2 E16.5 and 2 E18.5). On the other hand, E16.5 (e.g. Figure 5.5(c)) and E18.5 were more closely related

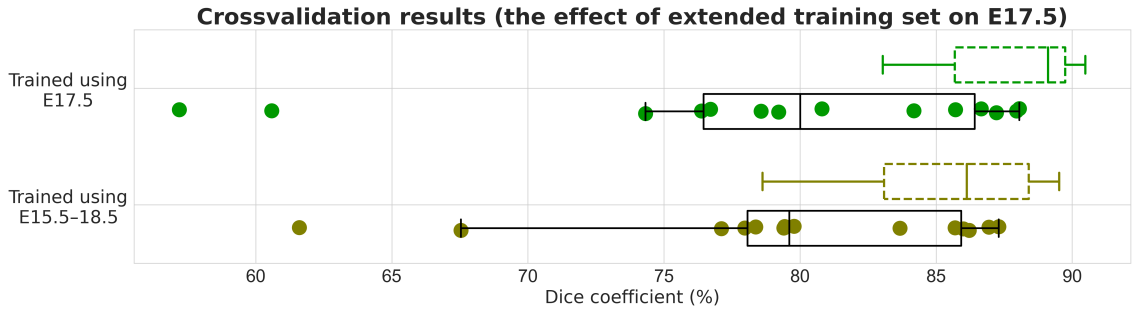


Figure 5.3: 7-fold crossvalidation results using the extended dataset. The boxplots for each experiment were derived using E17.5 samples in the same way as in Figure 5.1. The solid boxplot represents the results on the validation set, whereas the dashed boxplot reflects the training results.

to E17.5 [1] implying the acceptable results for these developmental stages, at least in case of wild types. Concerning the mutants (regardless of the developmental stage), it depended on a phenotype of the mutation. Naturally, very different-looking mutants (e.g. the E18.5 outlier, Figure 5.5(d)) had the worst results. Generally, the results for problematic samples could be improved by oversampling strategies or even better by acquiring additional data in the future.

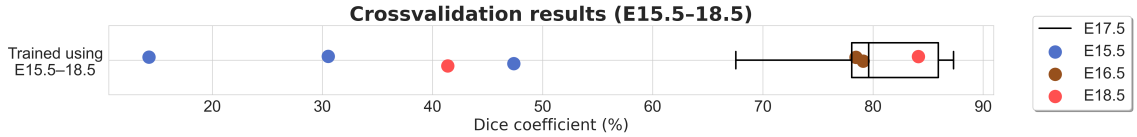
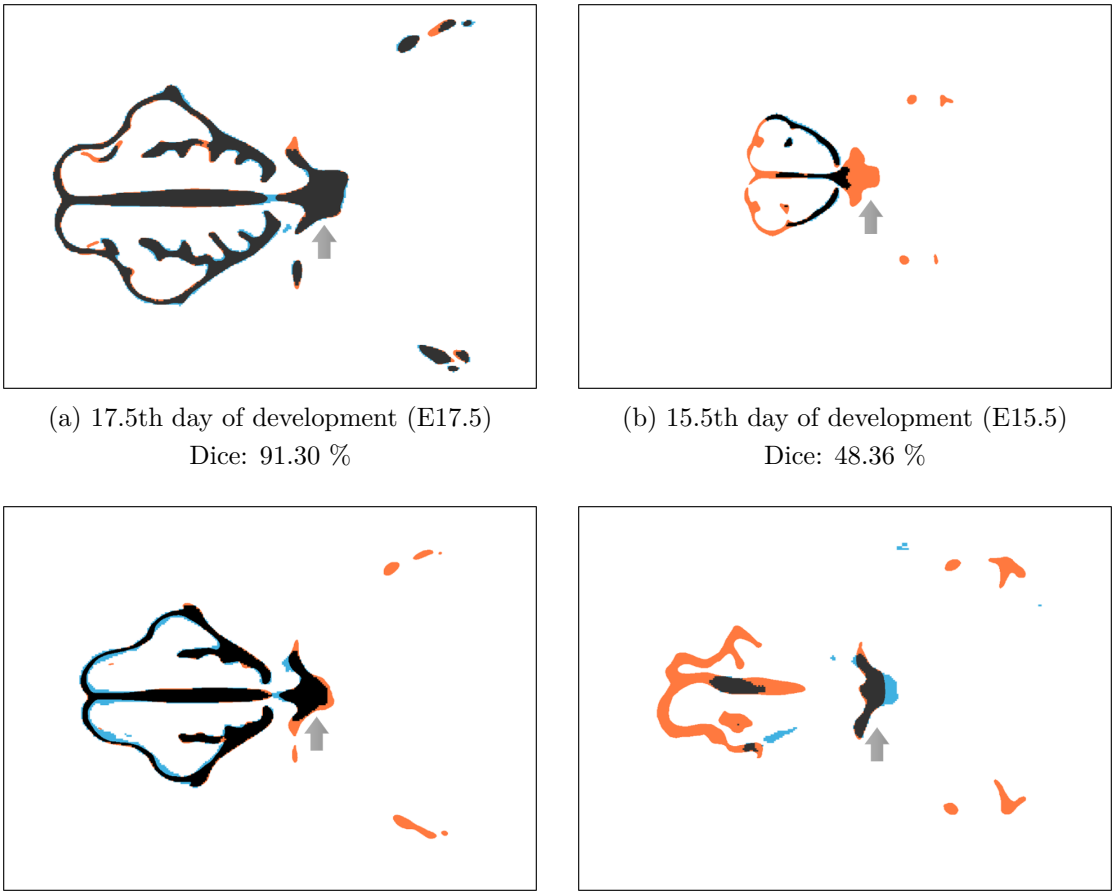


Figure 5.4: 7-fold crossvalidation results comparing the network's performance on various developmental stages.

5.4 Comparison with the previous research

Finally, moving back to "only E17.5" scenario, the proposed pipeline of the 3D piecewise segmentation was compared with a slice by slice segmentation from the previous research [42]. Again, exactly the same 7-fold crossvalidation was performed for both approaches using the same ground truth. From Figure 5.6 it is evident that a slice by slice segmentation outperformed the 3D piecewise segmentation meaning that the whole anatomical information in the slice is more important than adding surrounding slices. Specifically, the median of Dice coefficient using the slice by slice segmentation was 85.09 % whereas the 3D piecewise segmentation reached 80.01 % regarding the same metric. Also, when looking at Figure 5.6 closer, the rank of mice in terms of validation results is approximately the same in both cases. In other

[1] The developmental biologists examine the mouse development each half of a day as it is a fast process (21 days in total) [23].



(a) 17.5th day of development (E17.5)
Dice: 91.30 %

(b) 15.5th day of development (E15.5)
Dice: 48.36 %

(c) 16.5th day of development (E16.5)
Dice: 81.71 %

(d) very different-looking mutant
Dice: 38.57 %

Figure 5.5: The effect of developmental stage and phenotype on the segmentation performance. Black color represents TP, orange color FN, blue color FP. In the case of this figure, the Dice coefficient was computed only for the given slice (i.e. not for the whole volume as usual). To have better intuition about the appearance of particular developmental stages as well as a very different-looking mutant, corresponding slices were chosen according to a characteristic structure marked by the grey arrow.

words, concerning the automatic segmentation of the craniofacial cartilage, there are generally easier (in the right part of the boxplots) and more difficult (in the left part of the boxplots) samples. Interestingly, the additional surrounding slices rather confused the network in case of very different-looking mutant (*Red_24722e1*) as the slice by slice segmentation performed remarkably better on this sample. On the other hand, in the case of the overstained embryo (*Blue_12769-1++_Chd7g+Chd7+*) where the cartilage had an extraordinary range of voxel intensities, the surrounding slices greatly improve the segmentation performance.

The visual comparison of the 3D piecewise segmentation with a slice by slice segmentation is depicted in Figure 5.7 or in Figure 5.8. Evidently, the surround-

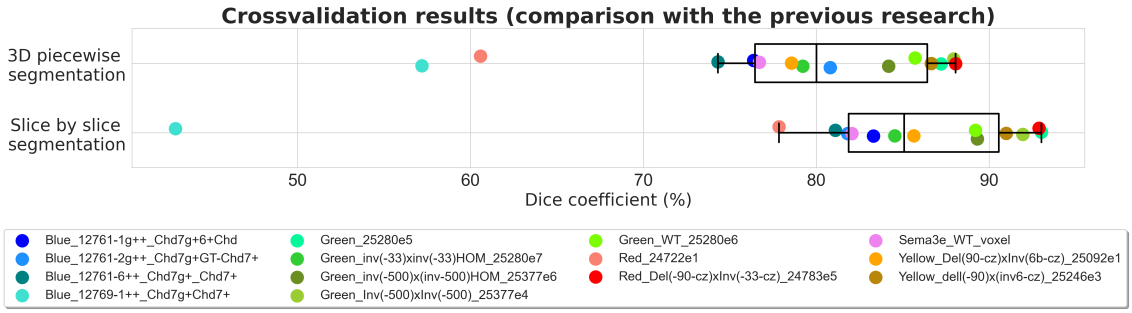
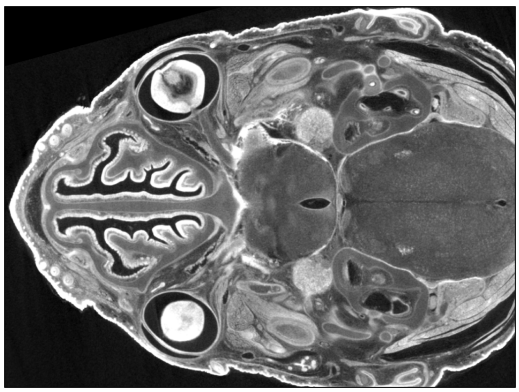


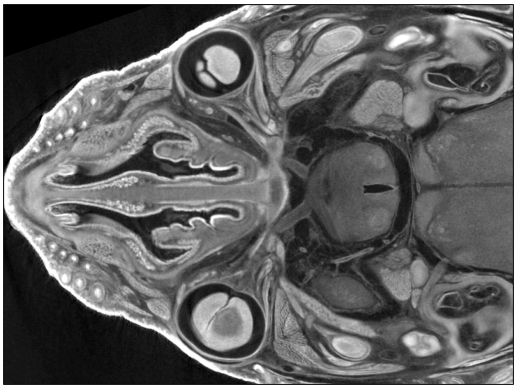
Figure 5.6: 7-fold crossvalidation results comparing the proposed network's performace with the previous research.

ing slices greatly helped in the case of the overstained embryo, however, in the case of properly prepared samples, the 3D information did not improve the results. Moreover, the proposed pipeline generated the segmentation prediction with square edges (Figure 5.8(b)), having a negative effect on the visual evaluation. On the other hand, the slice by slice segmentation suffered from discontinuities in other planes (Figure 5.8(c)) as it performed the segmentation just in one plane.

Generally speaking, the upsampling of the segmentation prediction to the original resolution was probably the weakest part of the proposed pipeline. Of course, processing the 3D image data in the original resolution would output the best results as no details would be lost by downsampling, but since the data were memory-intensive the downsampling was inevitable in the preprocessing phase. The nearest neighbour upsampling in the postprocessing phase was naturally not able to restore the fine details lost by downsampling resulting in the segmentation prediction with square edges. Smoother segmentation prediction in the original resolution could be obtained using transposed convolution in an extended decoder as proposed in the previous research [42], however, this approach was too memory-intensive for the 3D case. Also, the segmentation prediction could be smoothed e.g. by 3D Gaussian filter with subsequently applied thresholding for binarization of the mask. Nonetheless, this approach would naturally lead to inaccuracies e.g. due to the filling of small gaps in the mask which should remain unfilled. Theoretically speaking, the downsampled segmentation prediction could be converted from a voxelized form to a triangular mesh form and scaled to the desired size while mitigating the square edges. Also, the model of the craniofacial cartilage could be directly 3D printed using the triangular mesh representation. Still, however, the model's resolution would be dependent on the factor of downsampling used in the preprocessing phase.

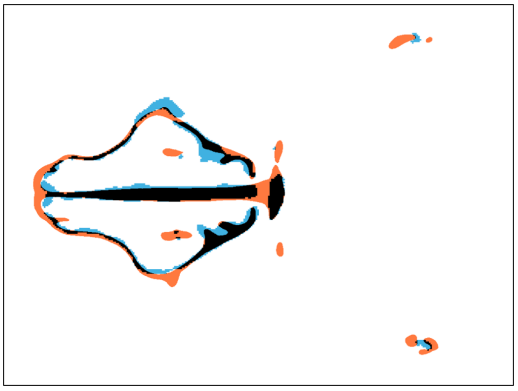


(a) properly prepared embryo

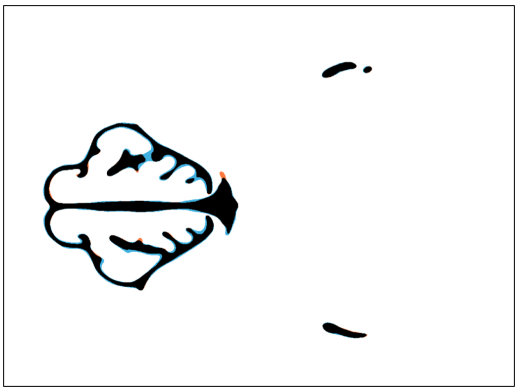


(d) overstained embryo

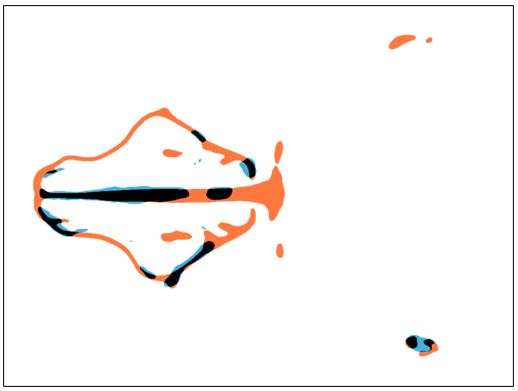
(b) 3D piecewise segmentation
Dice: 88.97 %



(e) 3D piecewise segmentation
Dice: 62.70 %



(c) slice by slice segmentation
Dice: 90.60 %



(f) slice by slice segmentation
Dice: 45.48 %

Figure 5.7: Comparison of the proposed methodology with the previous research. Specifically, two example slices (a, d) are depicted in the separate columns along with the segmentation prediction using a 3D piecewise segmentation (b, d) or a slice by slice segmentation (c, f). Black color represents TP, orange color FN, blue color FP. In the case of this figure, the Dice coefficient was computed only for the given slice (i.e. not for the whole volume as usual). Note: in (a) the craniofacial cartilage is darker than its neighborhood, whereas in (d) the cartilage has approximately equal intensity or it is even brighter than its neighborhood, which is caused by overstaining the embryo.

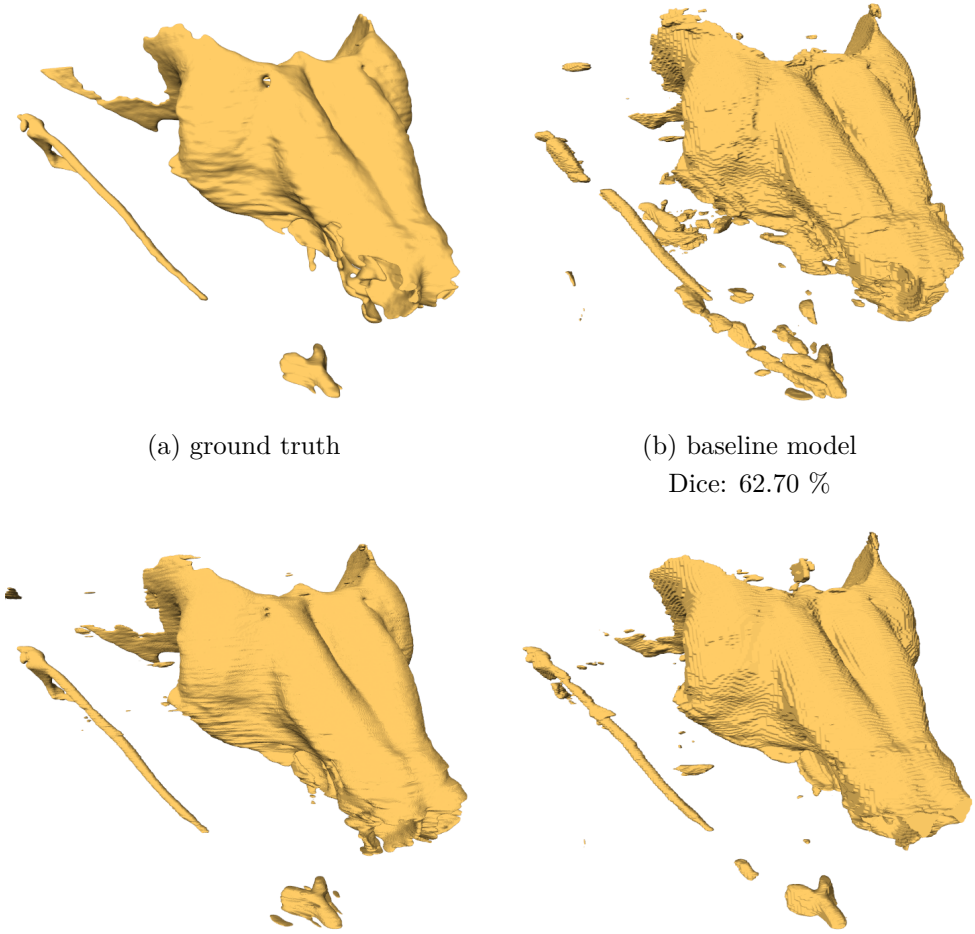
(a) ground truth (b) 3D piecewise segmentation (c) slice by slice segmentation

Figure 5.8: Comparison of the proposed methodology with the previous research (weaknesses of both approaches). In (a) there is a ground truth binary mask for a zoomed area of a given *sagittal* slice. In (b) it can be seen that the weakness of 3D piecewise segmentation are square edges in all planes including the sagittal. In (c) it is evident that there are several discontinuities (green arrows) in the sagittal segmentation prediction which is a natural weakness of the slice by slice segmentation using only the axial slices.

5.5 Overall evaluation

The main results of the master thesis can be subjectively evaluated in Figure 5.9. Firstly, the proposed optimization process greatly improved the segmentation performance of the 3D piecewise segmentation. Thus, using these advanced architectural modifications in further research is highly encouraged. Secondly, on the given segmentation problem, the slice by slice segmentation performed better than the 3D piecewise segmentation. It can be assumed that the square edges generated by the proposed pipeline decreased the results a bit, however, this could not be the reason for the approximately 5% drop in comparison with the slice by slice segmentation. The main problem seems to be the need for dividing the micro-CT data into the patches resulting in the loss of global contextual information. On the other hand, concerning the slice by slice segmentation, the network could benefit from preserving the relationships between particular anatomical structures as it used the whole slice instead of patches.

3D CNNs can be a very powerful tool for segmentation of small compact objects of interest in 3D image data, however, in the case of the oblong structures (such as the craniofacial cartilage) they are not able to boost the results since the 3D information cannot be fully exploited. Even though the pipeline of the 3D piecewise segmentation could be further tuned, the results of the master thesis rather support continuing with 2D CNNs in further research as the 3D spatial information of the features did not overcome the already mentioned disadvantages of the 3D piecewise segmentation. The 3D information can be incorporated into the segmentation either by using a stack of slices as the input to 2D CNNs or using the network



(a) ground truth

(b) baseline model
Dice: 62.70 %

(c) 2D U-Net
Dice: 85.64 %

(d) optimized model
Dice: 78.57 %

Figure 5.9: 3D models depicting the effect of the optimization process (b → d) as well as the comparison of the proposed pipeline (d) with the previous research (c). Dice coefficient was computed for the whole volume as usual.

with several input branches each processing the slices from a different plane. Also, the slice by slice segmentation can be further tuned, e.g. by using the proposed optimization process applicable to any U-Net-like architecture. Of course, other specialized blocks can be added to the architecture in the future such as the attention module [49], classification-guided module [24], etc. Next, tuning the hyperparameters of learning is as much important as tuning the CNN architecture. For instance, an insertion of several warm restarts [39] in the learning rate schedule can be exploited in further research. Last but not least, the general pipeline can be extended, e.g. by advanced contrast enhancement of the 3D image data or by incorporating further augmentations including the local geometrical ones imitating the dysmorphogenesis of the embryos.

Conclusion

At the beginning of the master thesis, the reader was introduced into the problem of the mouse embryo craniofacial cartilage segmentation. In the theoretical part, X-ray micro computed tomography (micro-CT) was described as the imaging modality used for scanning the embryos. Also, the reader was acquainted with convolutional neural networks (CNNs) as well as the training procedure along with its difficulties. Additionally, various attitudes proposed in the literature were described regarding the 3D image segmentation using CNNs.

In the practical part, on the basis of available data and hardware, the 3D piecewise segmentation pipeline was proposed aiming to incorporate the 3D information (naturally contained in micro-CT data) into the segmentation process. The reader was acquainted with necessary preprocessing and postprocessing steps and mainly with the processing part based on the 3D CNN. Specifically, the baseline model inspired by V-Net was proposed as well as its learning based on the current trends in the literature. The baseline model was further optimised with an emphasis on the utilization of advanced architectural modifications (ASPP, SELU, multi-output supervision, Dense blocks) presented in the literature in recent years.

Finally, 7-fold crossvalidation was performed for various experiments in the evaluation part. Firstly, it was shown that the optimization process raises the median of Dice coefficient from 69.74 % to 80.01 % and remarkably reduces an interquartile range of the results. Secondly, several non-architectural steps in the proposed methodology were justified. Thirdly, the experiments with other available developmental stages than E17.5 were described and evaluated. Lastly, the proposed solution was compared with the previous research. According to the results, in the case of the oblong structures (such as the craniofacial cartilage), a slice by slice segmentation performs better than the 3D piecewise segmentation. In the future, the proposed optimization process can be easily applied in U-Net from the previous research and probably improve the benchmark for the given segmentation problem.

Bibliography

[1] ASGARI TAGHANAKI, Saeid et al. Deep semantic segmentation of natural and medical images: a review. *Artificial Intelligence Review* [online]. 2020 [cit. 2020-12-30]. ISSN 0269-2821, 1573-7462. http://link.springer.com/10.1007/s10462-020-09854-1

[2] ATALOGLOU, Dimitrios et al. Fast and Precise Hippocampus Segmentation Through Deep Convolutional Neural Network Ensembles and Transfer Learning. *Neuroinformatics* [online]. 2019, **17**(4), 563–582. ISSN 1559-0089. https://link.springer.com/article/10.1007/s12021-019-09417-y

[3] BADEA, C. T. et al. In vivosmall-animal imaging using micro-CT and digital subtraction angiography. *Physics in Medicine and Biology* [online]. 2008, **53**(19), R319–R350 [cit. 2020-10-28]. ISSN 0031-9155. https://iopscience.iop.org/article/10.1088/0031-9155/53/19/R01

[4] BADRINARAYANAN, V., A. KENDALL & R. CIPOLLA. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* [online]. 2017, **39**(12), 2481–2495 [cit. 2020-12-30]. ISSN 1939-3539. https://ieeexplore.ieee.org/document/7803544

[5] BEVILACQUA, Vitoantonio et al. A comparison between two semantic deep learning frameworks for the autosomal dominant polycystic kidney disease segmentation based on magnetic resonance images. *BMC Medical Informatics and Decision Making* [online]. 2019, **19**(Suppl 9). [cit. 2020-12-29]. ISSN 1472-6947. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6907104/

[6] BONMATI, Ester et al. Automatic segmentation method of pelvic floor levator hiatus in ultrasound using a self-normalizing neural network. *Journal of Medical Imaging* [online]. 2018, **5**(2) [cit. 2021-04-15]. ISSN 2329-4302. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5762003/

[7] BROWNLEE, Jason. A Gentle Introduction to the Rectified Linear Unit (ReLU). *Machine Learning Mastery* [online]. Melbourne, 2019 [cit. 2020-12-12]. https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/

[8] BROWNLEE, Jason. Random Oversampling and Undersampling for Imbalanced Classification. *Machine Learning Mastery* [online]. Melbourne, 2019 [cit. 2020-12-31]. https://machinelearningmastery.com/random-oversampling-and-undersampling-for-imbalanced-classification/

[9] BURDI, Alphonse R. Developmental Biology and Morphogenesis of the Face, Lip and Palate. In: *Cleft Lip and Palate* [online]. Berlin/Heidelberg: Springer-Verlag, 2006, p. 3–12 [cit. 2020-11-09]. ISBN 978-3-540-23409-8. http://link.springer.com/10.1007/3-540-30020-1_1

[10] CASTRO, D. et al. Towards Optimizing Convolutional Neural Networks for Robotic Surgery Skill Evaluation. In: *2019 International Joint Conference on Neural Networks (IJCNN)* [online]. 2019, p. 1–8. [cit. 2021-04-15]. ISSN 2161-4407. https://ieeexplore.ieee.org/document/8852341

[11] CHEN, Jinghui et al. Closing the Generalization Gap of Adaptive Gradient Methods in Training Deep Neural Networks. In: *International Joint Conferences on Artificial Intelligence 2020* [online]. 2020 [cit. 2021-04-16]. http://arxiv.org/abs/1806.06763

[12] CHEN, Liang-Chieh et al. DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. *IEEE Transactions on Pattern Analysis and Machine Intelligence* [online]. 2018, **40**(4), 834–848. [cit. 2020-12-30]. ISSN 0162-8828, 2160-9292. http://ieeexplore.ieee.org/document/7913730/

[13] CHEN, Liang-Chieh et al. Rethinking Atrous Convolution for Semantic Image Segmentation. 2017 [cit. 2021-05-01]. http://arxiv.org/abs/1706.05587

[14] ÇIÇEK, Özgün et al. 3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation. In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2016* [online]. Cham: Springer International Publishing, 2016, p. 424–432. [cit. 2020-11-30]. ISBN 978-3-319-46723-8. https://link.springer.com/chapter/10.1007%2F978-3-319-46723-8_49

[15] CLEVERT, Djork-Arné, Thomas UNTERTHINER & Sepp HOCHREITER. Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). In *4th International Conference on Learning Representations, ICLR 2016* [online]. 2016 [cit. 2020-12-31]. http://arxiv.org/abs/1511.07289

[16] DU PLESSIS, Anton et al. Laboratory x-ray micro-computed tomography: a user guideline for biological samples. *GigaScience* [online]. 2017, **6**(6) [cit. 2020-10-21]. ISSN 2047-217X. https://academic.oup.com/gigascience/article/doi/10.1093/gigascience/gix027/3737665

[17] GAO, Yunhe et al. Multi-resolution Path CNN with Deep Supervision for Intervertebral Disc Localization and Segmentation. In: *Medical Image Computing and Computer Assisted Intervention – MICCAI 2019* [online]. Cham:

Springer International Publishing, 2019, p. 309–317 [cit. 2021-03-29]. ISBN 978-3-030-32245-8. https://link.springer.com/chapter/10.1007/978-3-030-32245-8_35

[18] GLOROT, Xavier & Yoshua BENGIO. Understanding the difficulty of training deep feedforward neural networks. *Journal of Machine Learning Research* [online]. 2010. [cit. 2020-12-15]. http://proceedings.mlr.press/v9/glorot10a/glorot10a.pdf

[19] GOODFELLOW, Ian, Yoshua BENGIO & Aaron COURVILLE. *Deep Learning* [online]. MIT Press, 2016, p. 224–270, 271–325, 326–366 [cit. 2020-11-21]. http://www.deeplearningbook.org

[20] HE, Kaiming et al. Deep Residual Learning for Image Recognition. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* [online]. 2016, p. 770–778. ISSN 1063-6919. [cit. 2020-12-15]. https://ieeexplore.ieee.org/document/7780459

[21] HE, Kaiming et al. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In: *2015 IEEE International Conference on Computer Vision (ICCV)* [online]. Santiago, Chile: IEEE, 2015, p. 1026–1034 [cit. 2020-12-15]. ISBN 978-1-4673-8391-2. https://www.cv-foundation.org/openaccess/content_iccv_2015/papers/He_Delving_Deep_into_ICCV_2015_paper.pdf

[22] HERMANEK, Petr et al. Principles of X-ray Computed Tomography. In: Simone CARMIGNATO, Wim DEWULF & Richard LEACH, ed. *Industrial X-Ray Computed Tomography* [online]. Cham: Springer International Publishing, 2018, p. 25–67 [cit. 2020-10-21]. ISBN 978-3-319-59573-3. https://link.springer.com/chapter/10.1007/978-3-319-59573-3_2

[23] HILL, M. A. Embryology: Mouse Timeline Detailed. 2018 [cit. 2021-05-05]. https://embryology.med.unsw.edu.au/embryology/index.php/Mouse_Timeline_Detailed

[24] HUANG, H. et al. UNet 3+: A Full-Scale Connected UNet for Medical Image Segmentation. In: *2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* [online]. 2020, p. 1055–1059 [cit. 2020-04-17]. ISSN 2379-190X. https://ieeexplore.ieee.org/document/9053405

[25] IOFFE, Sergey a Christian SZEGEDY. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In: *32nd International Conference on Machine Learning, ICML 2015* [online]. 2015. [cit. 2020-12-15] ISBN 9781510810587. http://proceedings.mlr.press/v37/ioffe15.pdf

[26] JEGOU, Simon et al. The One Hundred Layers Tiramisu: Fully Convolutional DenseNets for Semantic Segmentation. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* [online]. Honolulu, HI, USA: IEEE, 2017, p. 1175–1183 [cit. 2020-12-25]. ISBN 978-1-5386-0733-6. https://ieeexplore.ieee.org/document/8014890

[27] KAMNITSAS, Konstantinos et al. DeepMedic for Brain Tumor Segmentation. In: *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries* [online]. Cham: Springer International Publishing, 2016, p. 138–149. Lecture Notes in Computer Science [cit. 2020-12-28]. ISBN 978-3-319-55524-9. https://link.springer.com/chapter/10.1007/978-3-319-55524-9_14

[28] KAUCKA, Marketa et al. Oriented clonal cell dynamics enables accurate growth and shaping of vertebrate cartilage. *eLife* [online]. 2017, 6, e25902 [cit. 2020-11-09]. ISSN 2050-084X. https://elifesciences.org/articles/25902

[29] KAUCKA, Marketa et al. Signals from the brain and olfactory epithelium control shaping of the mammalian nasal capsule cartilage. *eLife* [online]. 2018, 7, e34465 [cit. 2020-11-09]. ISSN 2050-084X. https://elifesciences.org/articles/34465

[30] KESKAR, Nitish Shirish & Richard SOCHER. Improving Generalization Performance by Switching from Adam to SGD. [online]. 2017 [cit. 2021-04-16]. http://arxiv.org/abs/1712.07628

[31] KINGMA, Diederik P. & BA, Jimmy Lei. Adam: A method for stochastic optimization. In: *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings* [online]. 2015. [cit. 2020-12-15]. https://arxiv.org/pdf/1412.6980

[32] KLAMBAUER, Günter et al. Self-Normalizing Neural Networks. In: *Advances in Neural Information Processing Systems* [online]. 2017, 30, 972-981 [cit. 2021-04-15]. http://arxiv.org/abs/1706.02515

[33] KOČENDOVÁ, Kateřina. *Automatické vyhlazení 3D modelů kraniální embryonální myší chrupavky* [online]. Brno, 2020. Diplomová práce (Ing.).

Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav biomedicínského inženýrství. Vedoucí práce: Ing. Roman Jakubíček. [cit. 2020-11-14]. https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=208640

[34] LANDIS, Eric N. & Denis T. KEANE. X-ray microtomography. *Materials Characterization* [online]. 2010, **61**(12), 1305–1316. ISSN 10445803. [cit. 2020-10-21]. https://linkinghub.elsevier.com/retrieve/pii/S1044580310002706

[35] LIU, Liyuan et al. On the Variance of the Adaptive Learning Rate and Beyond. In: *International Conference on Learning Representations (ICLR) 2020* [online]. 2020 [cit. 2021-04-29]. https://arxiv.org/abs/1908.03265

[36] LIU, Xiaolong, Zhidong DENG & Yuhan YANG. Recent progress in semantic image segmentation. *Artificial Intelligence Review* [online]. 2019, **52**(2), 1089–1106. ISSN 1573-7462. [cit. 2020-11-22]. https://link.springer.com/article/10.1007/s10462-018-9641-3

[37] LONG, Jonathan, Evan SHELHAMER & Trevor DARRELL. Fully convolutional networks for semantic segmentation. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* [online]. 2015, p. 3431–3440 [cit. 2020-12-30]. ISSN 1063-6919. https://ieeexplore.ieee.org/document/7298965

[38] LOSHCHILOV, Ilya & Frank HUTTER. Decoupled Weight Decay Regularization. In: *International Conference on Learning Representations (ICLR) 2019* [online]. 2019 [cit. 2021-04-16]. https://arxiv.org/abs/1711.05101.arXiv:1711.05101

[39] LOSHCHILOV, Ilya & Frank HUTTER. SGDR: Stochastic Gradient Descent with Warm Restarts. In: *International Conference on Learning Representations (ICLR) 2017* [online]. 2017 [cit. 2021-05-09]. http://arxiv.org/abs/1608.03983

[40] MA, Xiangyuan et al. U-Net-based Deep-Learning Bladder Segmentation in CT Urography. *Medical physics* [online]. 2019, **46**(4), 1752–1765 [cit. 2021-03-29]. ISSN 0094-2405. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6453730/

[41] MAAS, Andrew L., Hannun Y. AWNI & Andrew Y. NG. Rectifier Nonlinearities Improve Neural Network Acoustic Models. *Proceedings of the*

30th International Conference on Machine Learning [online]. Atlanta, 2013. [cit. 2020-12-31]. http://ai.stanford.edu/~amaas/papers/relu_hybrid_icml2013_final.pdf

[42] MATULA, Jan. *Segmentace chrupavčité tkáně ve 3D mikro CT snímcích myších embryí* [online]. Brno, 2019, Diplomová práce (Ing.). Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav biomedicínského inženýrství. Vedoucí práce: Ing. Jiří Chmelík. [cit. 2020-11-11]. https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=190599

[43] METSCHER, Brian D. MicroCT for comparative morphology: simple staining methods allow high-contrast 3D imaging of diverse non-mineralized animal tissues. *BMC Physiology* [online]. 2009, 9, 11. ISSN 1472-6793. [cit. 2020-12-05]. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2717911/

[44] MILLETARI, F., N. NAVAB & S. AHMADI. V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation. In: *2016 Fourth International Conference on 3D Vision (3DV)* [online]. 2016, p. 565–571. [cit. 2020-11-30]. ISSN 2378-3826, 2475-7888. https://ieeexplore.ieee.org/document/7785132

[45] MLYNARSKI, Pawel et al. 3D convolutional neural networks for tumor segmentation using long-range 2D context. *Computerized Medical Imaging and Graphics* [online]. 2019, 73, 60–72. ISSN 08956111. [cit. 2020-12-29]. https://hal.inria.fr/hal-01883716v2/document

[46] MIZUTANI, Ryuta & Yoshio SUZUKI. X-ray microtomography in biology. *Micron* [online]. 2012, **43**(2–3), 104–115. [cit. 2020-10-21]. ISSN 09684328. https://linkinghub.elsevier.com/retrieve/pii/S0968432811001788

[47] MORTAZI, Aliasghar et al. CardiacNET: Segmentation of Left Atrium and Proximal Pulmonary Veins from MRI Using Multi-view CNN. In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2017* [online]. Cham: Springer International Publishing, 2017. [cit. 2020-12-29]. ISBN 978-3-319-66185-8. https://link.springer.com/chapter/10.1007/978-3-319-66185-8_43

[48] NOH, H., S. HONG & B. HAN. Learning Deconvolution Network for Semantic Segmentation. In: *2015 IEEE International Conference on Computer Vision (ICCV)* [online]. 2015, p. 1520–1528. [cit. 2020-12-30]. ISSN 2380-7504. https://ieeexplore.ieee.org/document/7410535

[49] OKTAY, Ozan et al. Attention U-Net: Learning Where to Look for the Pancreas. In: *1st Conference on Medical Imaging with Deep Learning (MIDL 2018)* [online]. Amsterdam, 2018 [cit. 2020-12-30]. https://arxiv.org/abs/1804.03999

[50] OLLION, Jean et al. TANGO: a generic tool for high-throughput 3D image analysis for studying nuclear organization. *Bioinformatics* [online]. 2013, **29**(14), 1840–1841. ISSN 1460-2059, 1367-4803. [cit. 2020-11-15]. https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/btt276

[51] RONNEBERGER, Olaf, Philipp FISCHER & Thomas BROX. U-Net: Convolutional Networks for Biomedical Image Segmentation. In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015* [online]. Cham: Springer International Publishing, 2015, p. 234–241. Lecture Notes in Computer Science [cit. 2020-12-30]. ISBN 978-3-319-24574-4.https://link.springer.com/chapter/10.1007%2F978-3-319-24574-4_28

[52] ROTH, Holger R. et al. Hierarchical 3D fully convolutional networks for multi-organ segmentation [online]. 2017 [cit. 2020-12-29]. http://arxiv.org/abs/1704.06382

[53] SALEHI, Seyed Sadegh Mohseni, Deniz ERDOGMUS & Ali GHOLIPOUR. Tversky Loss Function for Image Segmentation Using 3D Fully Convolutional Deep Networks. In: *Machine Learning in Medical Imaging* [online]. Cham: Springer International Publishing, 2017, p. 379–387. Lecture Notes in Computer Science [cit. 2021-04-25]. ISBN 978-3-319-67389-9. https://link.springer.com/chapter/10.1007/978-3-319-67389-9_44

[54] SIMONYAN, Karen & Andrew ZISSERMAN. Very Deep Convolutional Networks for Large-Scale Image Recognition. In: *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings* [online]. 2015 [cit. 2020-12-11]. http://arxiv.org/abs/1409.1556

[55] STOCK, Stuart R. *Microcomputed tomography: methodology and applications.* Boca Raton: CRC Press, 2009, p. 39–84. ISBN 978-1-4200-5876-5.

[56] Tensorflow. Distributed training with TensorFlow. *Tensorflow* [online]. ©2015–2021 [cit. 2021-04-25]. https://www.tensorflow.org/guide/distributed_training

[57] Tensorflow. tf.py_fuction. *Tensorflow* [online]. ©2015–2021 [cit. 2021-04-29]. https://www.tensorflow.org/versions/r2.2/api_docs/python/tf/py_function

[58] TESAŘOVÁ, M. et al. Use of micro computed-tomography and 3D printing for reverse engineering of mouse embryo nasal capsule. *Journal of Instrumentation* [online]. 2016, **11**(03), C03006–C03006 [cit. 2020-10-21]. ISSN 1748-0221. https://iopscience.iop.org/article/10.1088/1748-0221/11/03/C03006

[59] WILSON, Ashia C. et al. The Marginal Value of Adaptive Gradient Methods in Machine Learning. In: *31st Conference on Neural Information Processing Systems (NIPS 2017)* [online]. 2017 [cit. 2021-04-16]. http://arxiv.org/abs/1705.08292

[60] WU, Yuxin & Kaiming HE. Group Normalization. *International Journal of Computer Vision* [online]. 2018. [cit. 2020-12-15]. https://arxiv.org/abs/1803.08494

[61] XUE, Yunzhe et al. A multi-path 2.5 dimensional convolutional neural network system for segmenting stroke lesions in brain MRI images. *NeuroImage: Clinical* [online]. 2020, 25, 102118. ISSN 22131582. [cit. 2020-12-29]. https://www.sciencedirect.com/science/article/pii/S2213158219304656

[62] YU, Fisher & Vladlen KOLTUN. Multi-Scale Context Aggregation by Dilated Convolutions. In: *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings* [online]. 2016. [cit. 2020-12-11]. http://arxiv.org/abs/1511.07122

[63] ZENG, Guodong et al. 3D U-net with Multi-level Deep Supervision: Fully Automatic Segmentation of Proximal Femur in 3D MR Images. In: *Machine Learning in Medical Imaging* [online]. Cham: Springer International Publishing, 2017, p. 274–282. [cit. 2021-05-02] ISBN 978-3-319-67389-9. https://link.springer.com/chapter/10.1007/978-3-319-67389-9_32

[64] ZHAO, Hengshuang et al. Pyramid Scene Parsing Network. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* [online]. Honolulu, HI: IEEE, 2017, p. 6230–6239 [cit. 2020-12-30]. ISBN 978-1-5386-0457-1. http://ieeexplore.ieee.org/document/8100143/

[65] ZHOU, Zongwei et al. UNet++: A Nested U-Net Architecture for Medical Image Segmentation. In: *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support* [online]. Cham: Springer International Publishing, 2018, p. 3–11. Lecture Notes in Computer Science [cit. 2020-04-17]. ISBN 978-3-030-00889-5. https://link.springer.com/chapter/10.1007/978-3-030-00889-5_1

List of symbols, quantities and abbreviations

2D, 3D, 4D	2-Dimensional, 3-Dimensional, 4-Dimensional
ASPP	Atrous Spatial Pyramid Pooling
API	Application Programming Interface, i.e. set of available interactions with a given software
CPU	Central Processing Unit, i.e. main processor of a computer performing various tasks
CNN	Convolutional Neural Network
GPU	Graphics Processing Unit, i.e. processor with highly parallelized structure performing a specialized task
HDD	Hard Disk Drive, i.e. older type of computer storage disk
micro-CT	X-ray micro Computed Tomography
PReLU	Parametric Rectified Linear Unit
RAM	Random Access Memory, i.e. readable and changeable computer memory
SELU	Scaled Exponential Linear Unit
SSD	Solid State Drive, i.e. newer type of computer storage disk

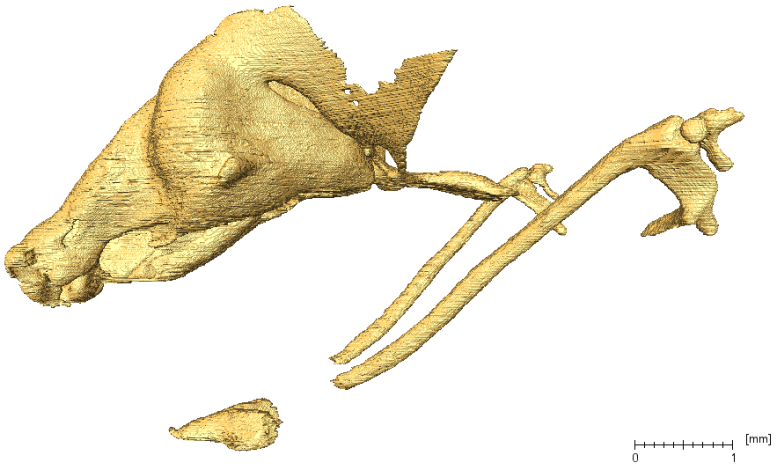
List of appendices

A Details about the available data

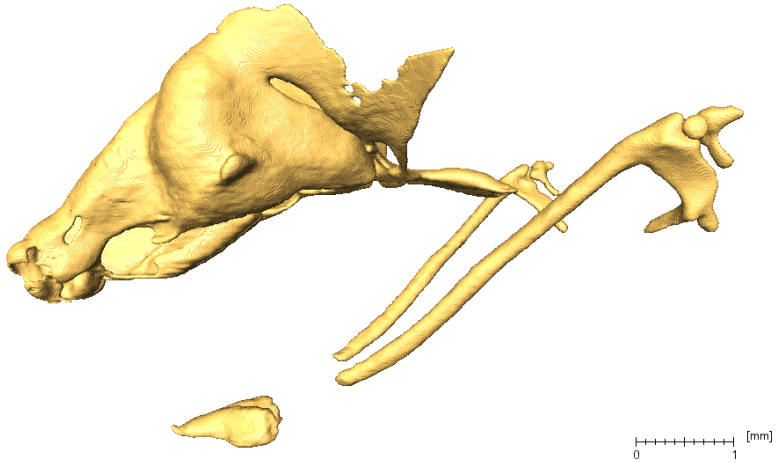
Table A.1: Available data.

Dataset name	Dimensions	Voxel size (µm)	Developmental stage	Wild type or mutant
Ascl_1mut_CreERT2_E165	$1418 \times 1877 \times 1279$	5.0	E16.5	mutant
Ascl-CreERT-E165-ctrl	$1050 \times 1579 \times 1003$	6.0	E16.5	wild type
Blue_12761-1g++_Chd7g+6+Chd	$1158 \times 1637 \times 1257$	6.2	E17.5	wild type
Blue_12761-2g++_Chd7g+GT-Chd7+	$1478 \times 2073 \times 1379$	5.7	E17.5	wild type
Blue_12761-6++_Chd7g+_Chd7+	$1372 \times 1936 \times 1436$	6.0	E17.5	wild type
Blue_12769-1++_Chd7g+Chd7+	$1267 \times 1946 \times 1386$	6.0	E17.5	wild type
Col2DTA_E155_mut_5037_1	$824 \times 1100 \times 968$	6.7	E15.5	mutant
Col2DTA_E155_mut_5037_6	$789 \times 1084 \times 848$	6.7	E15.5	mutant
Green_25280e5	$1262 \times 1906 \times 1462$	5.6	E17.5	wild type
Green_inv(-33)xinv(-33)HOM_25280e7	$1340 \times 1854 \times 1395$	5.0	E17.5	mutant
Green_inv(-500)x(inv-500)HOM_25377e6	$1346 \times 1786 \times 1409$	5.0	E17.5	mutant
Green_Inv(-500)xInv(-500)_25377e4	$1244 \times 1783 \times 1158$	5.3	E17.5	wild type
Green_WT_25280e6	$1344 \times 1801 \times 1344$	5.8	E17.5	wild type
Red_24722e1	$1408 \times 1886 \times 1536$	4.5	E17.5	mutant
Red_Del(-90-cz)xInv(-33-cz)_24783e5	$1406 \times 1961 \times 1429$	5.2	E17.5	wild type
Sema3e_WT_voxel	$1990 \times 2314 \times 1530$	4.9	E17.5	wild type
SHH_E15_5_ctrl	$1053 \times 1271 \times 2031$	6.0	E15.5	wild type
SIX_SIX++_WT_E185	$1473 \times 1981 \times 1438$	5.5	E18.5	wild type
Six1−_E185	$1069 \times 1642 \times 1116$	5.5	E18.5	mutant
Yellow_Del(90-cz)xInv(6b-cz)_25092e1	$1351 \times 1926 \times 1441$	5.7	E17.5	wild type
Yellow_dell(-90)x(inv6-cz)_25246e3	$1350 \times 1934 \times 1589$	5.0	E17.5	wild type

B Effect of the median filtering on a binary mask

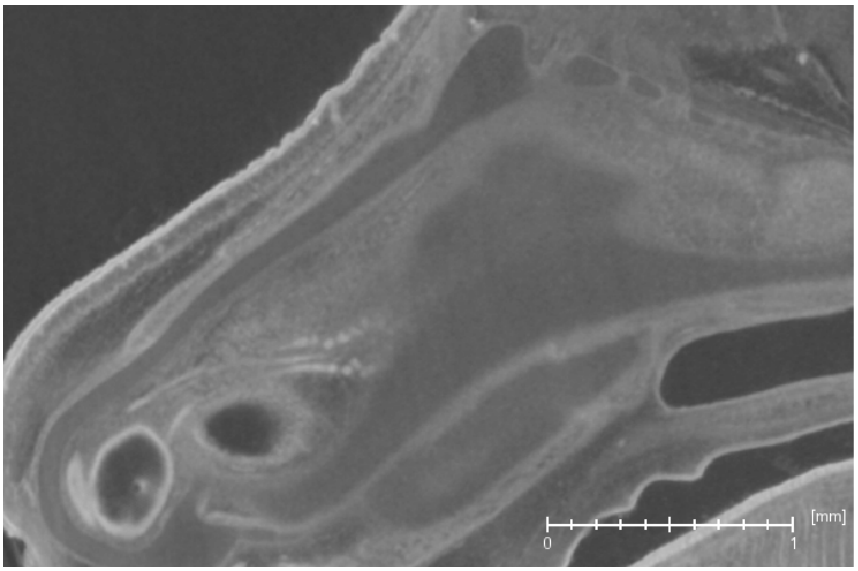


(a)

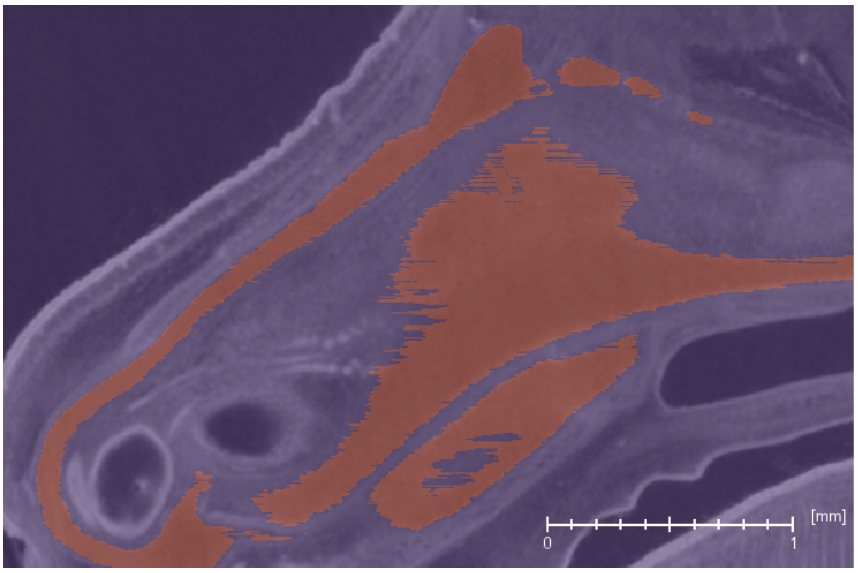


(b)

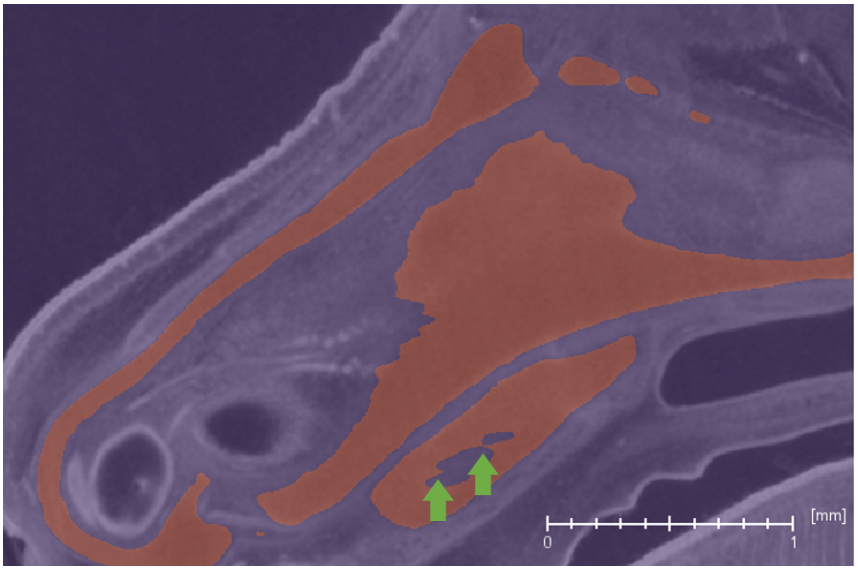
Figure B.1: Effect of the median filtering with $5 \times 5 \times 5$ kernel on a binary mask (two-page figure). A 3D model of the craniofacial cartilage generated from the original (a) and smoothed (b) binary mask. Also, on the detail of the chosen slice (c), it is evident that a contour of the craniofacial cartilage (red) is noisy (d) which limits the profit of using the 3D information. After smoothing (e), the contours are more acceptable, even though a few inaccuracies (e.g. green arrows) may occur. All images are viewed from the sagittal perspective.



(c)



(d)



(e)

C Comparison of Adam and AdamW optimizer

Considering parameters θ of the network, the weight decay can be incorporated into the SGD optimization process as

$$\theta_{t+1} = \theta_t - \alpha \cdot \nabla f_t(\theta_t) - \lambda \cdot \theta_t \tag{C.1}$$

where $\nabla f_t(\theta_t)$ is the *t-th* mini-batch gradient, α is a learning rate and λ defines the weight decay. [38]

In common deep learning libraries, L2 regularization is implemented instead of the weight decay since (in the case of SGD) it represents the same. Suppose L2 regularized loss function

$$f_t^{reg}(\theta_t) = f_t(\theta_t) + \frac{\lambda^{'}}{2} \cdot \|\theta_t\|_2^2 \tag{C.2}$$

and its gradient

$$\nabla f_t^{reg}(\theta_t) = \nabla f_t(\theta_t) + \lambda^{'} \cdot \theta_t \tag{C.3}$$

When defining the regularization term $\lambda^{'}$ as $\frac{\lambda}{\alpha}$ we can write

$$\theta_{t+1} = \theta_t - \alpha \cdot (\nabla f_t(\theta_t) + \frac{\lambda}{\alpha} \cdot \theta_t) \tag{C.4}$$

which can be rewritten as the equation eq. C.1. [38]

Since introducing SGD, several new optimizers have been developed, but L2 regularization remains as a classical way of implementing the weight decay. However, as the authors of AdamW pointed out, in the case of adaptive learning rate algorithms (such as Adam), the weight decay *cannot* be implemented through L2 regularization properly. To address this problem, they refine Adam to AdamW (Figure C.1). [38]

Specifically, the problem of Adam with L2 regularization is the fact that the parameters with bigger gradient magnitudes are less regularized [38]. This can be derived from Figure C.1. For simplicity, suppose a sufficient time step and therefore $\hat{m_t} = m_t$ and $\hat{v_t} = v_t$. Then, the optimization step in Adam with L2 regularization can be written as

$$\theta_t = \theta_{t-1} - \eta_t \cdot (\alpha \cdot \frac{\beta_1 \cdot m_{t-1} + (1-\beta_1) \cdot (\nabla f_t(\theta_{t-1}) + \lambda \cdot \theta_{t-1})}{\sqrt{v_t} + \epsilon}) \tag{C.5}$$

From the eq. C.5 it is evident that the regularization term λ is divided by $\sqrt{v_t}$. Thus, the bigger the parameter's gradient magnitude, the smaller the regularization of the parameter.

Algorithm 2 Adam with L_2 regularization and Adam with decoupled weight decay (AdamW)

given $\alpha = 0.001, \beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}, \lambda \in \mathbb{R}$
initialize time step $t \leftarrow 0$, parameter vector $\boldsymbol{\theta}_{t=0} \in \mathbb{R}^n$, first moment vector $\boldsymbol{m}_{t=0} \leftarrow \boldsymbol{0}$, second moment vector $\boldsymbol{v}_{t=0} \leftarrow \boldsymbol{0}$, schedule multiplier $\eta_{t=0} \in \mathbb{R}$
repeat
  $t \leftarrow t + 1$
  $\nabla f_t(\boldsymbol{\theta}_{t-1}) \leftarrow \text{SelectBatch}(\boldsymbol{\theta}_{t-1})$ ▷ select batch and return the corresponding gradient
  $\boldsymbol{g}_t \leftarrow \nabla f_t(\boldsymbol{\theta}_{t-1}) + \lambda \boldsymbol{\theta}_{t-1}$
  $\boldsymbol{m}_t \leftarrow \beta_1 \boldsymbol{m}_{t-1} + (1 - \beta_1) \boldsymbol{g}_t$ ▷ here and below all operations are element-wise
  $\boldsymbol{v}_t \leftarrow \beta_2 \boldsymbol{v}_{t-1} + (1 - \beta_2) \boldsymbol{g}_t^2$
  $\hat{\boldsymbol{m}}_t \leftarrow \boldsymbol{m}_t / (1 - \beta_1^t)$ ▷ β_1 is taken to the power of t
  $\hat{\boldsymbol{v}}_t \leftarrow \boldsymbol{v}_t / (1 - \beta_2^t)$ ▷ β_2 is taken to the power of t
  $\eta_t \leftarrow \text{SetScheduleMultiplier}(t)$ ▷ can be fixed, decay, or also be used for warm restarts
  $\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t-1} - \eta_t \left(\alpha \hat{\boldsymbol{m}}_t / (\sqrt{\hat{\boldsymbol{v}}_t} + \epsilon) + \lambda \boldsymbol{\theta}_{t-1} \right)$
until *stopping criterion is met*
return optimized parameters $\boldsymbol{\theta}_t$

Figure C.1: Comparison of Adam and AdamW optimizers (adopted from [38]) where g is a gradient of the loss function, m is a momentum term (exponential moving average of the past gradients), v is an adaptive learning rate term (exponential moving average of the past squared gradients), $\hat{m}$ and $\hat{v}$ are corrections of m and v for the early stage of learning, α is a learning rate, β_1 (or β_2) control the exponential decay rate of m (or v), ϵ is for numerical stability, and λ controls the amount of regularization.

On the other hand, AdamW implements the regularization using weight decay in the optimization step

$$\theta_t = \theta_{t-1} - \eta_t \cdot (\alpha \cdot \frac{\beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot \nabla f_t(\theta_{t-1})}{\sqrt{v_t} + \epsilon} + \lambda \cdot \theta_{t-1}) \tag{C.6}$$

resulting in equal regularization of all parameters and better generalization performance [38].

D Evaluation appendix

The removal of outliers did not influence the conclusion set in chapter 5 that states: Optimization process using advanced architectural modifications raised the median of Dice coefficient and also remarkably reduced the interquartile range of the results (Figure D.1). Specifically, when excluding the outliers, the median of Dice coefficient was raised from 73.65 % to 82.50 %.

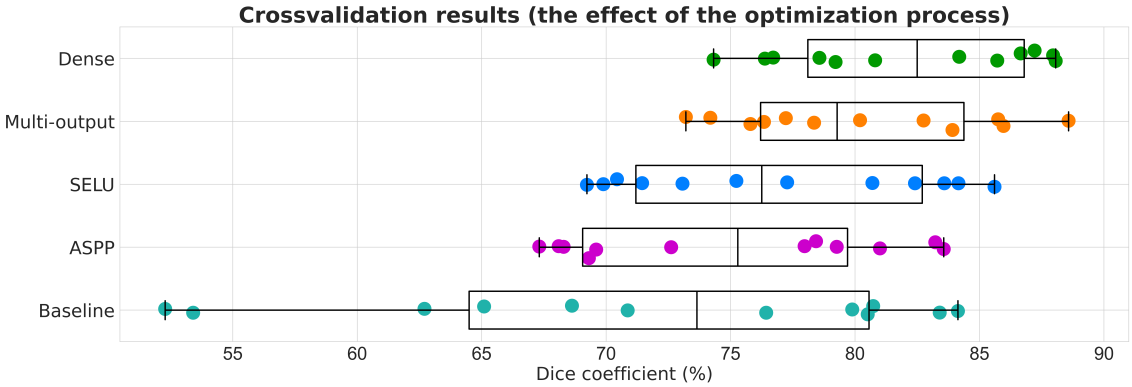


Figure D.1: 7-fold crossvalidation results mapping the optimization process (without outliers). For the description of the graph see Figure 5.1.

E Electronic attachements

VolumeModule.py contains the API for working with micro-CT data (class Volume) or subsequently its patches (class SubVolumeList).

MakeDataset.py is responsible for saving the patches on the disk.

DataToCNN.py implements the loading pipeline depicted in Figure 4.4.

Model.py defines several CNN models used in the evaluation of the proposed optimization process.

RunningModel.py is responsible for training the model as well as for obtaining the segmentation prediction for patches in the evaluation part.

Evaluation.py evaluates the CNN segmentation performance on the whole 3D image data.

The necessary packages can be installed in *Anaconda* using `conda env create -f master_thesis_Polakova.yml` command.